PROGRESS REPORT
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION
GRANT NAG 5-1129
CLUSTERS AND CORONAL MASS EJECTIONS

M. A. Coplan
August 4, 1994

Introduction

In the last 6 months of the grant period we have worked on the analysis of solar wind material originating from coronal holes. This analysis requires identification of equatorial holes on the surface of the sun and verification of the ICI data before analysis begins. To determine whether the composition of the solar wind originating from coronal holes is different from normal solar wind composition it is necessary to analyze ICI mass spectra for several days before and after the period of the coronal hole in addition to the coronal hole period. With from 30 to 50 mass spectra per day, as many as 700 spectra require analysis. Though this figure is often reduced by as much as a factor of 10 when there are data gaps and bad data, the analysis task is formidable. In order to automate the data analysis we have developed an data analysis manager program that eliminates the need for frequent human intervention and speeds analysis by two orders of magnitude. That which previously took days to complete now can be finished in hours. A description of the program is given below. Documentation including flow charts and code are included in the Appendix.

Data Manager Program

The main program used to manage the ICWIND data analysis program is JOEY.COM, but there are a number of associated files that must also be present for everything to function correctly. The following files must be present in the directory:

CHANGE_SETTINGS.COM
ICWIND2.COM
ICWIND_KB_JDN.EXC
ICWIND_KB_JDN.OBJ
SETTINGS.DAT

The following files should also be found in the directory, but are not essential for the program to functic

C.IND
DIRECTORIES.DAT

ICWIND_KB_JDN.FOR
READ_ME.DOC
REMOTE_SITE.DAT

The following files may also be found in the directory. They are generated by JOEY.COM but are not necessary to run it:

INPUT_FILE.DAT
INPUT_SCAN.DAT
OGILVIE.DAT
TIMES.DAT

Any files besides those listed above should be deleted form the program directory to minimize space used on the hard disk of the computer. Both CHANGE-SETTINGS.COM and JOEY.COM have extensive comments in them.

Descriptions of the programs along with flow charts and code are included in the Appendix.

Appendix

READ_ME.DOC

Thank you for taking the time to read this file. The main program used to manage the ICWIND data analysis progrm is JOEY.COM, but there are a number of associated files that must also be present for everything to function as advertised. For JOEY.COM to function properly, the following files MUST be present in this directory:

CHANGE_SETTINGS.COM	This companion program to JOEY.COM allows the user to change the parameters under which the ICWIND program operates.
ICWIND2.COM	Small program to put the FOR___.dat data files in the output directory.
ICWIND_KB_JDN.EXE	Executable file for ICWIND data analysis. I believe that KB stands for Kellie Bradach, a predecessor of mine who modified the ICWIND program extensively. I also modified the program slightly to remove the thirty-character filename restriction on the L, S, and P data files (the restriction is now sixty characters). Also, to fix an unidentified bug, I have (according to the instructions of Frank Ottens) commented out much of the SATDAT subroutine. The eliminated portion has to do with Versaplot, which we do not use anymore, anyway. To distinguish this version of the program from Kellie's original, I have added my initials to the name as well.
ICWIND_KB_JDN.OBJ	Fortran object file for the ICWIND program.
JOEY.COM	The manager program for all these files.
SETTINGS.DAT	Default settings for the ICWIND program.

As I said, the above files MUST all be present in the same directory for JOEY.COM to function. The following files SHOULD also be found in this directory, but are not essential to the program's function:

C.INC	Required file when compiling FORTRAN source code (.FOR files).
DIRECTORIES.DAT	Data file containing the default directories for the program files, the output data files, and the RWNAST input data file.
ICWIND_KB_JDN.FOR	Source code for ICWIND program (written in FORTRAN).
READ_ME.DOC	This file.
REMOTE_SITE.DAT	Stores default settings for a remote site to which the output data files are transferred.

The following files MAY also be found in the directory; they are generated by JOEY.COM, but are not necessary to its running:

ERROR.DAT	File that records errors during the running of ICWIND.
INPUT_FILE.DAT INPUT_SCAN.DAT	These are two files which JOEY.COM generates from which the ICWIND program runs.

READ_ME.DOC (second page)

OGILVIE.DAT	Data generated from running ICWIND from input_scan.dat alone--the Ogilvie option. This data is the time of each run, its lowest chi-squared, and the coronal temperature corresponding to that chi-squared. It all comes from FOR033.DAT.
TIMES.DAT	List of times corresponding to spectra to be analyzed.

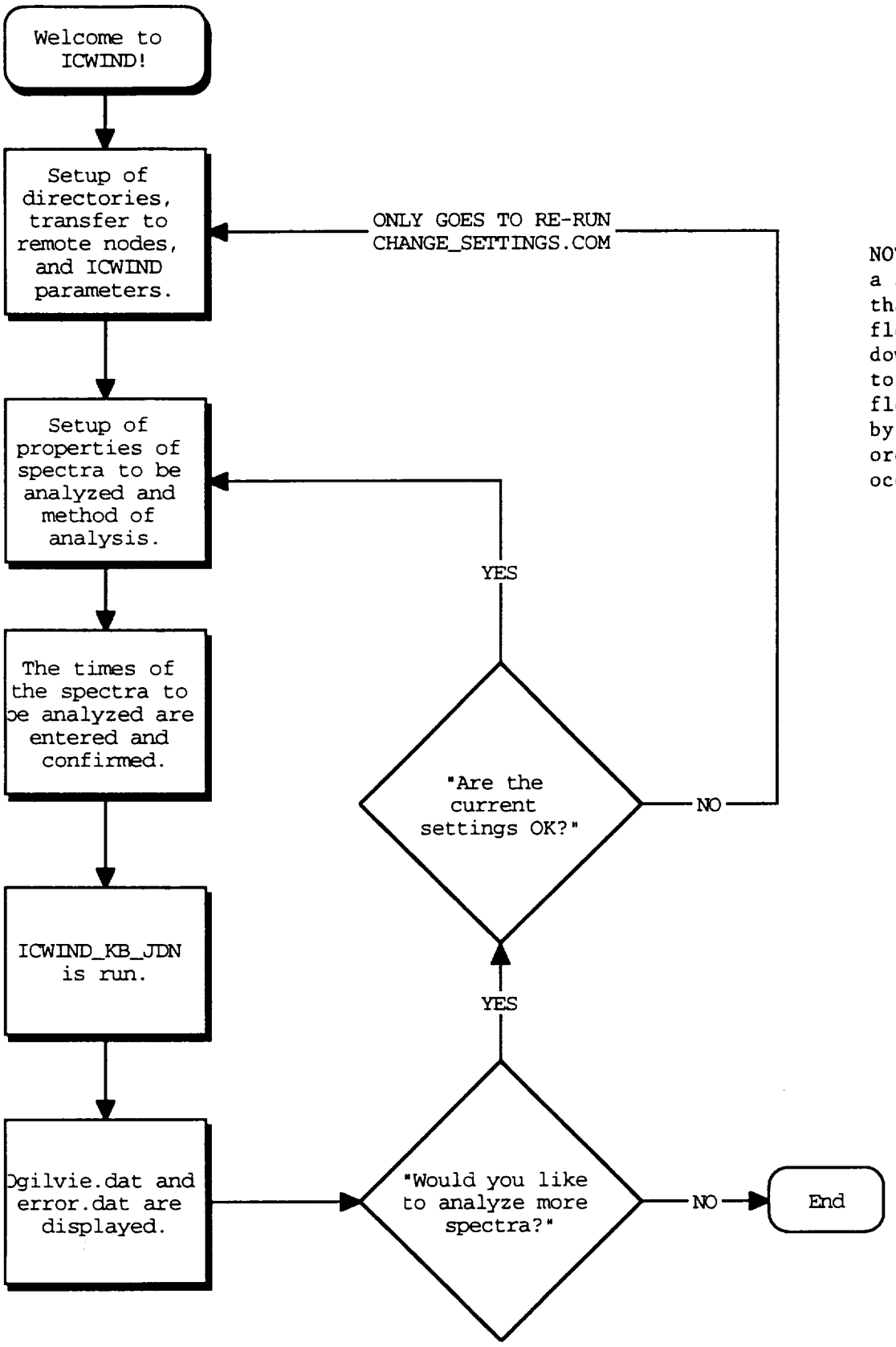
Any files besides those listed above should be promptly DELETED from the program directory to minimize disk clutter and possible confusion!

Both CHANGE_SETTINGS.COM and JOEY.COM have extensive comments in them, so if you have any question about either, please "type" them, and you will probably find the answer. I have tried to make my variable names as obvious as possible (and I usually append "$" to a name when it is a character string).

I hope you find these programs useful.

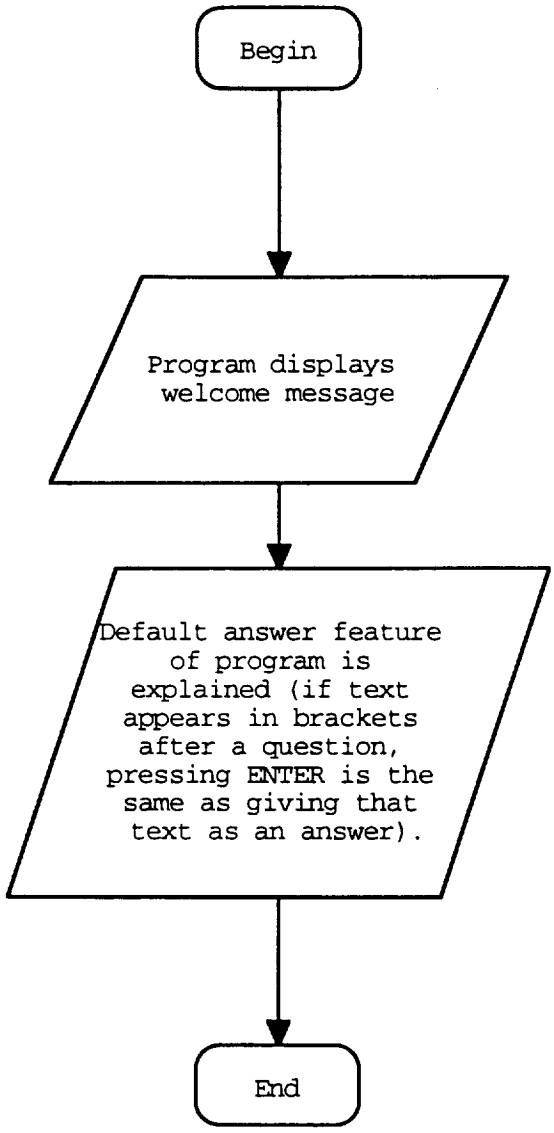
Joseph Newhouse
July 21, 1994

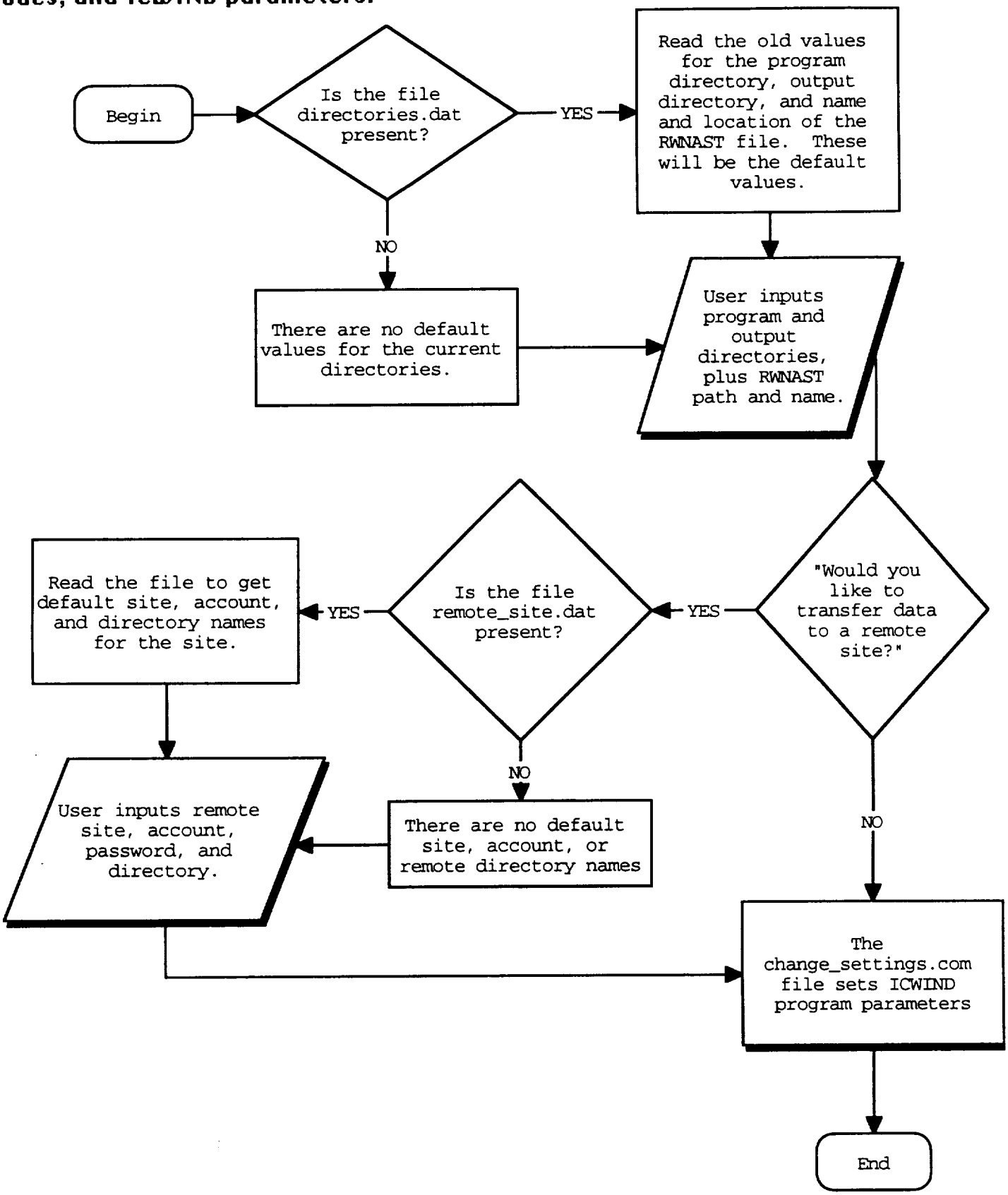
Flowchart of JOEY.COM **Thu, Jul 21, 1994**

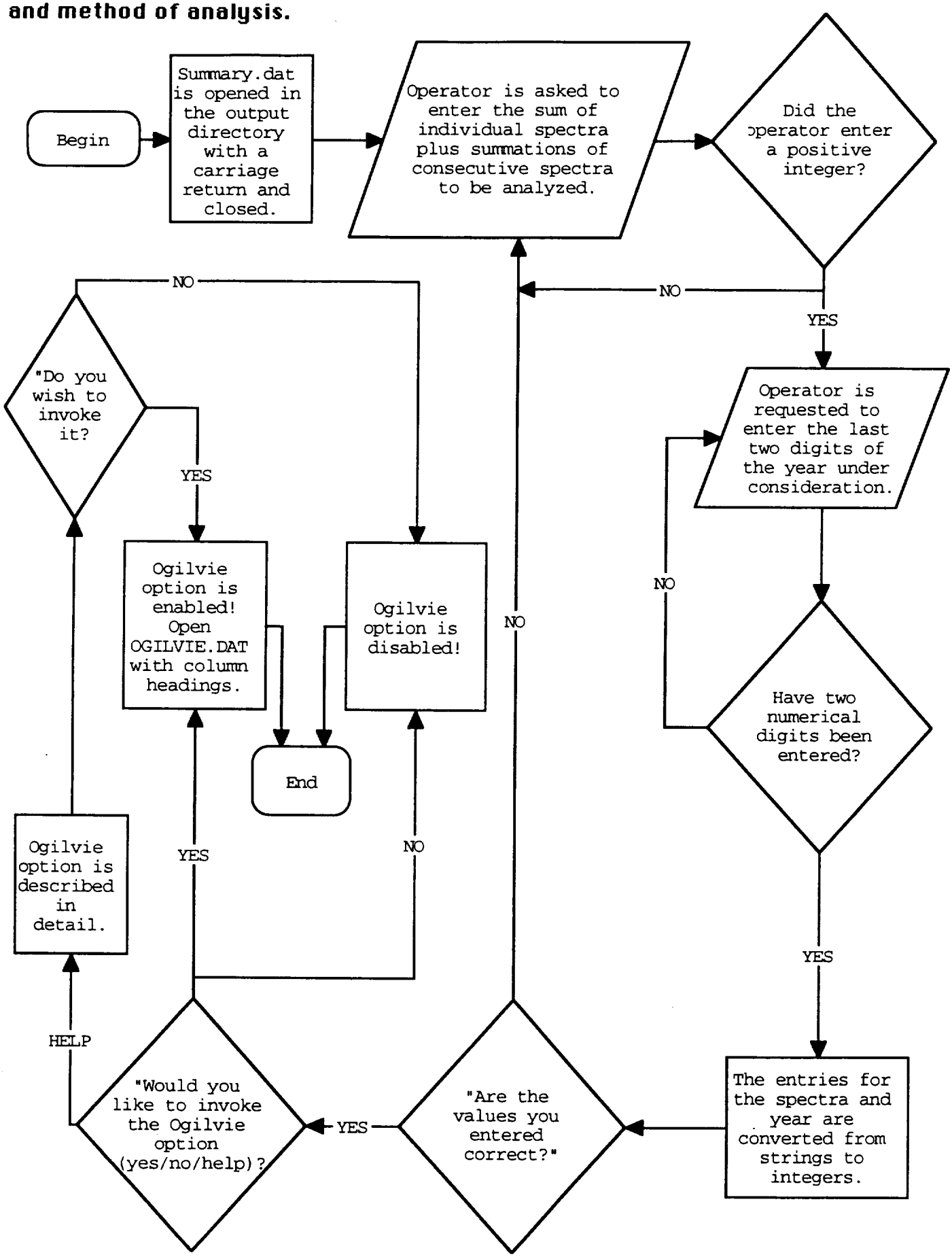


NOTE: The presence of a shadowed box indicates that there is a flowchart one level down corresponding to that box. All flowcharts are arranged by level, in the order in which they occur in the program.

TOP LEVEL

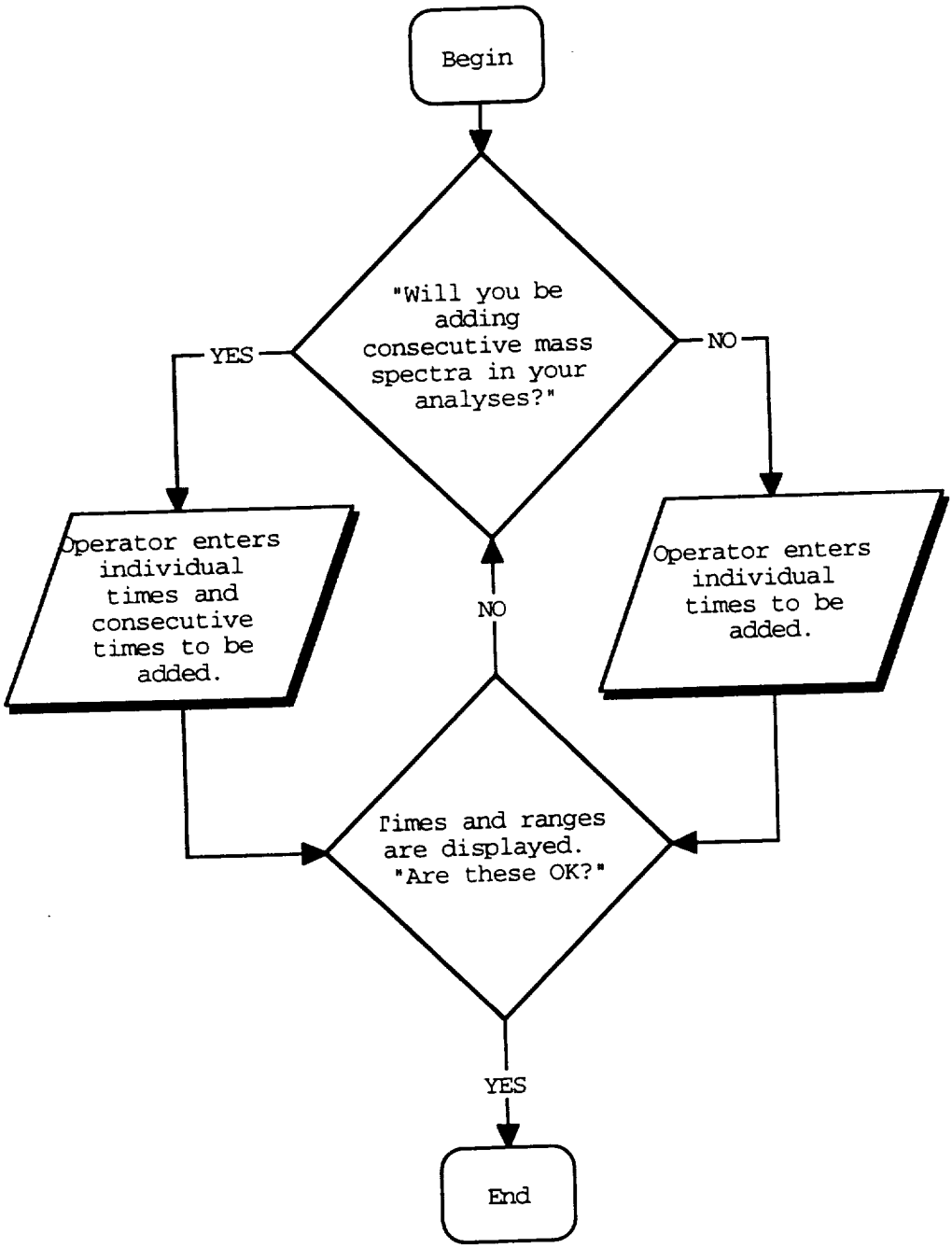

Begin
Program displays welcome message
Default answer feature of program is explained (if text appears in brackets after a question, pressing ENTER is the same as giving that text as an answer).
End

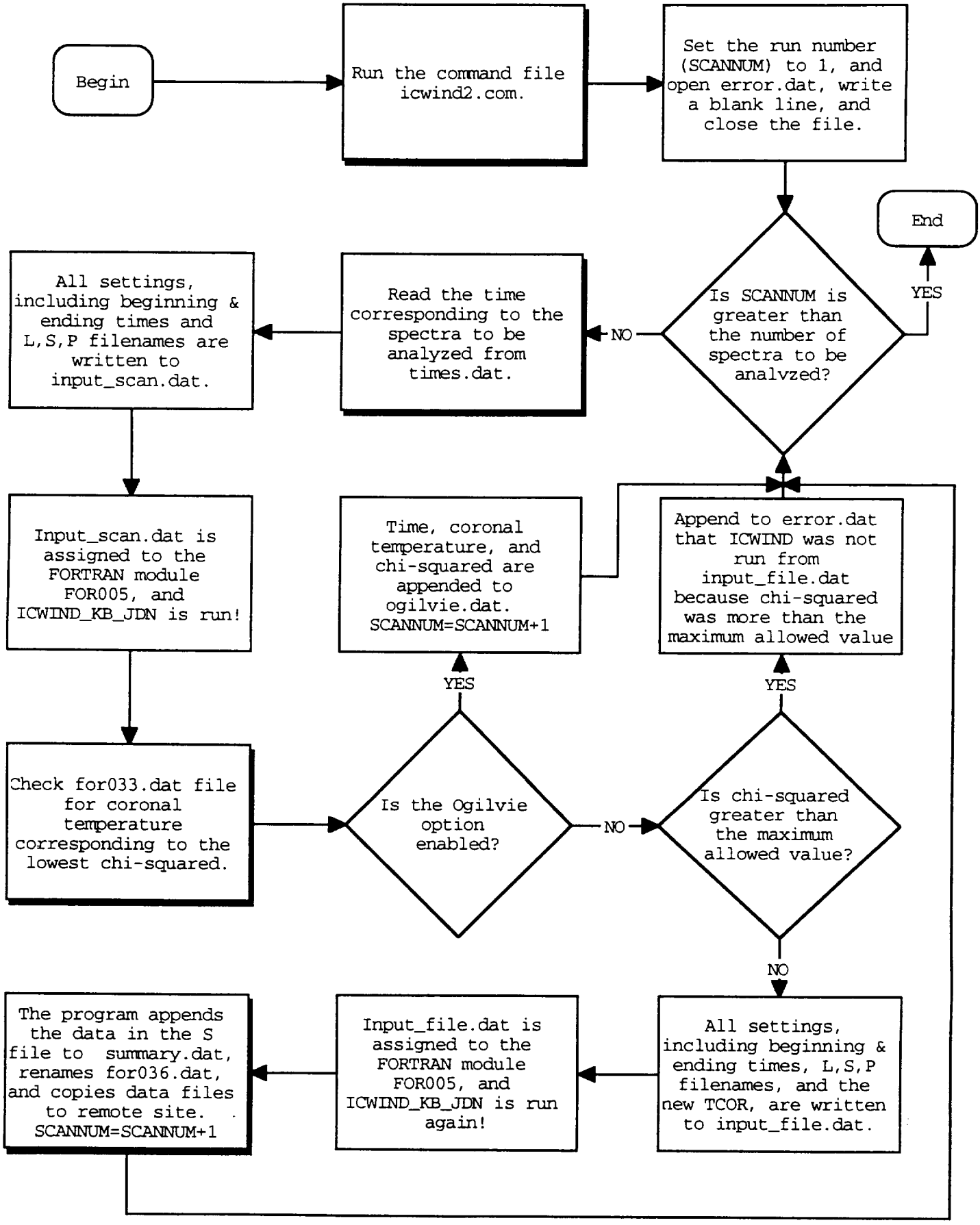

Begin
Is the file directories.dat present?
YES
Read the old values for the program directory, output directory, and name and location of the RWNAST file. These will be the default values.
NO
There are no default values for the current directories.
User inputs program and output directories, plus RWNAST path and name.
"Would you like to transfer data to a remote site?"
YES
Is the file remote_site.dat present?
YES
Read the file to get default site, account, and directory names for the site.
NO
There are no default site, account, or remote directory names
User inputs remote site, account, password, and directory.
NO
The change_settings.com file sets ICWIND program parameters
End

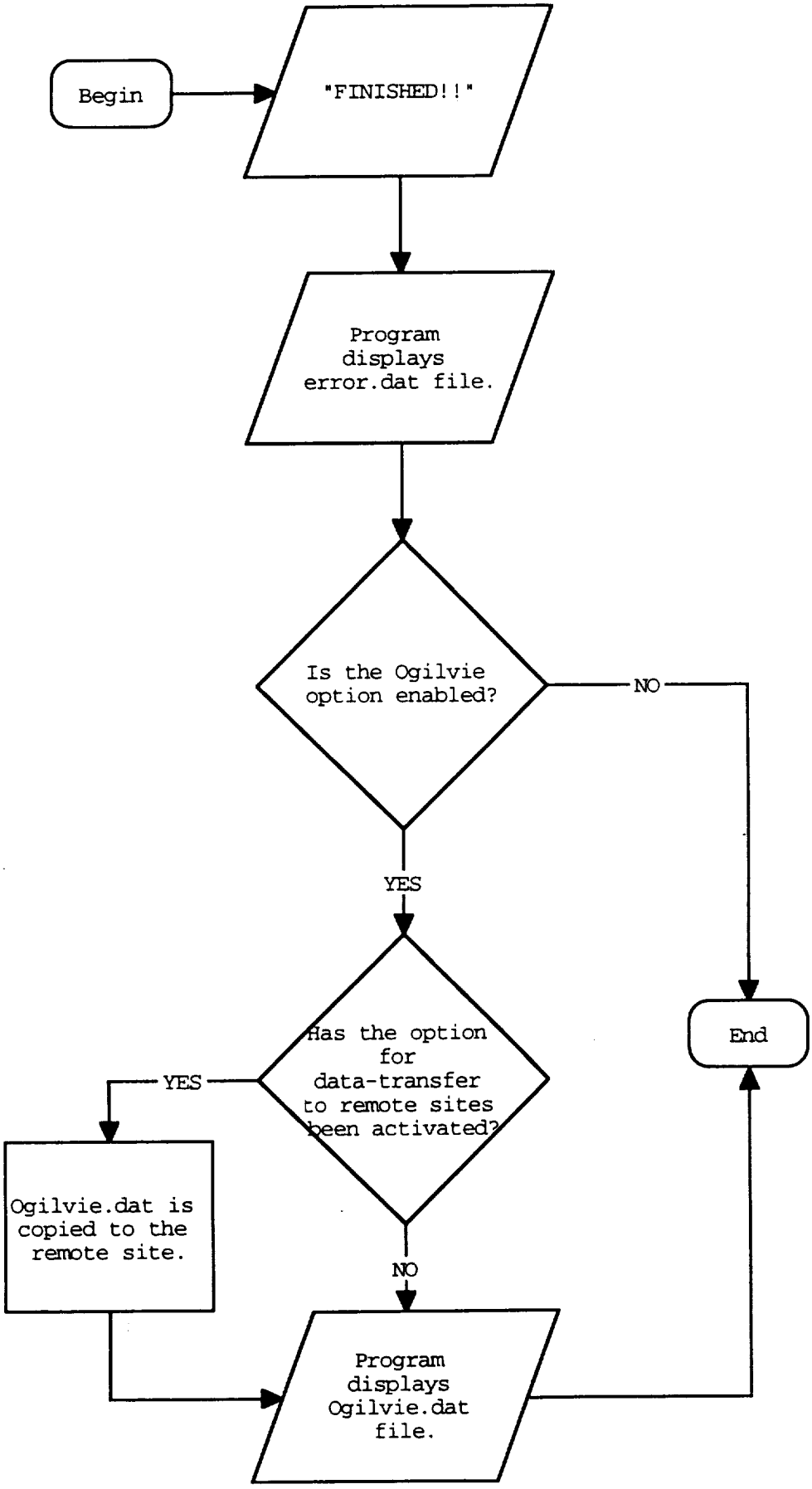


SECOND LEVEL

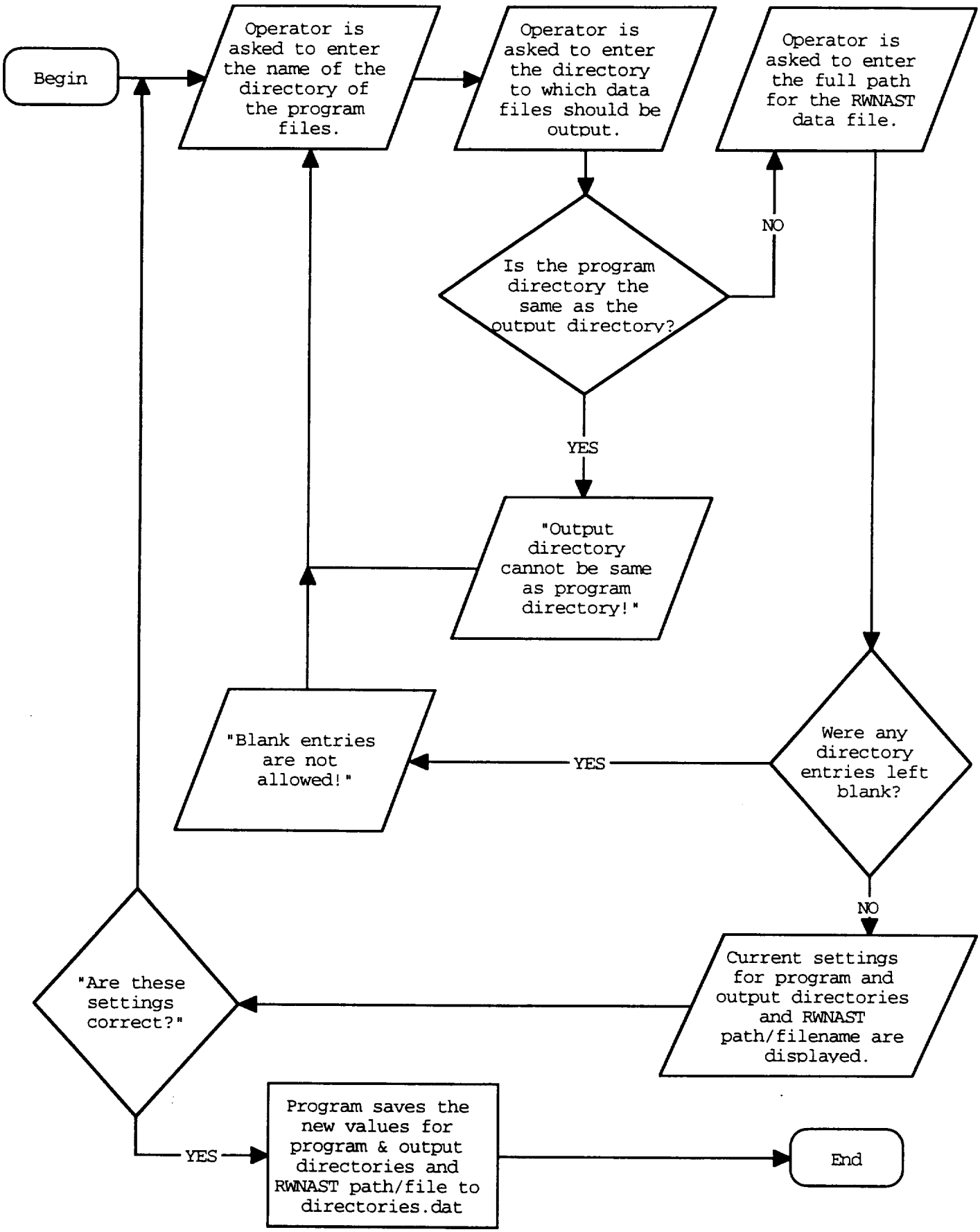
The times of the spectra to be analyzed are entered and confirmed.

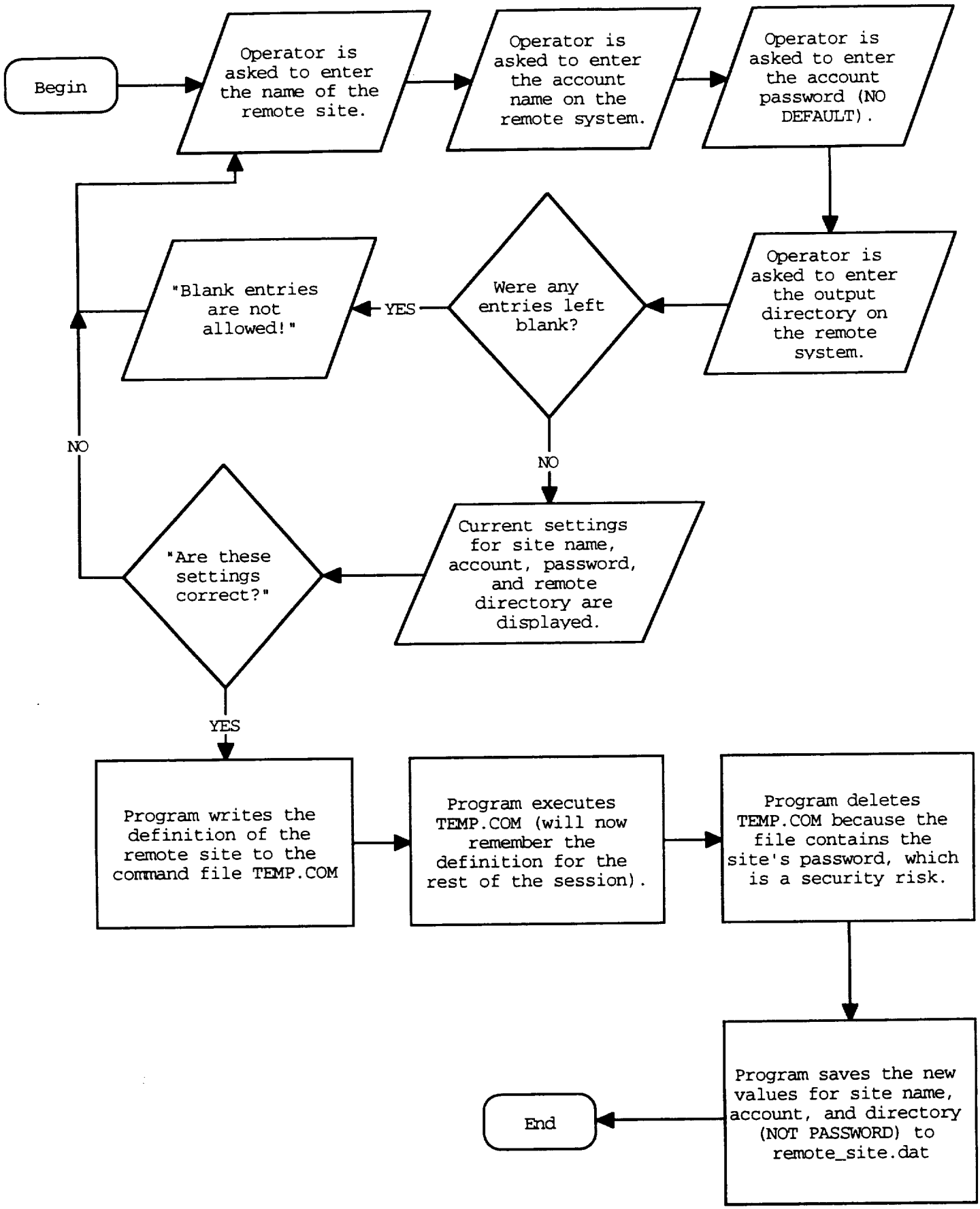


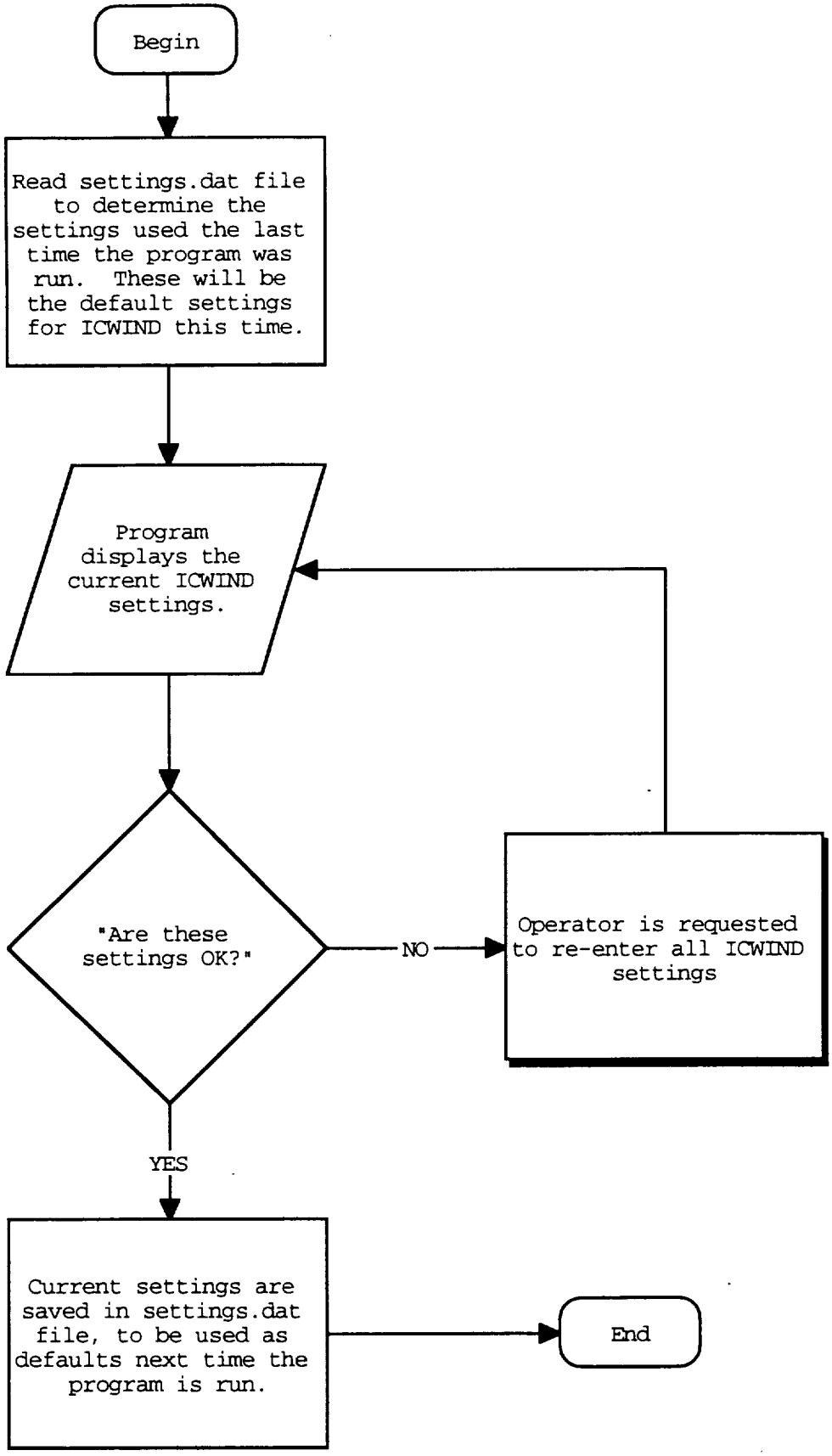

Begin
Run the command file icwind2.com.
Set the run number (SCANNUM) to 1, and open error.dat, write a blank line, and close the file.
End
YES
Is SCANNUM is greater than the number of spectra to be analvzed?
NO
Read the time corresponding to the spectra to be analyzed from times.dat.
All settings, including beginning & ending times and L,S,P filenames are written to input_scan.dat.
Input_scan.dat is assigned to the FOR005, and ICWIND_KB_JDN is run!
Check for033.dat file for coronal temperature corresponding to the lowest chi-squared.
Is the Ogilvie option enabled?
YES
Time, coronal temperature, and chi-squared are appended to ogilvie.dat. SCANNUM=SCANNUM+1
Is chi-squared greater than the maximum allowed value?
YES
Append to error.dat that ICWIND was not run from input_file.dat because chi-squared was more than the maximum allowed value
NO
All settings, including beginning & ending times, L,S,P filenames, and the new TCOR, are written to input_file.dat.
Input_file.dat is assigned to the FORTRAN module FOR005, and ICWIND_KB_JDN is run again!
The program appends the data in the S file to summary.dat, renames for036.dat, and copies data files to remote site. SCANNUM=SCANNUM+1

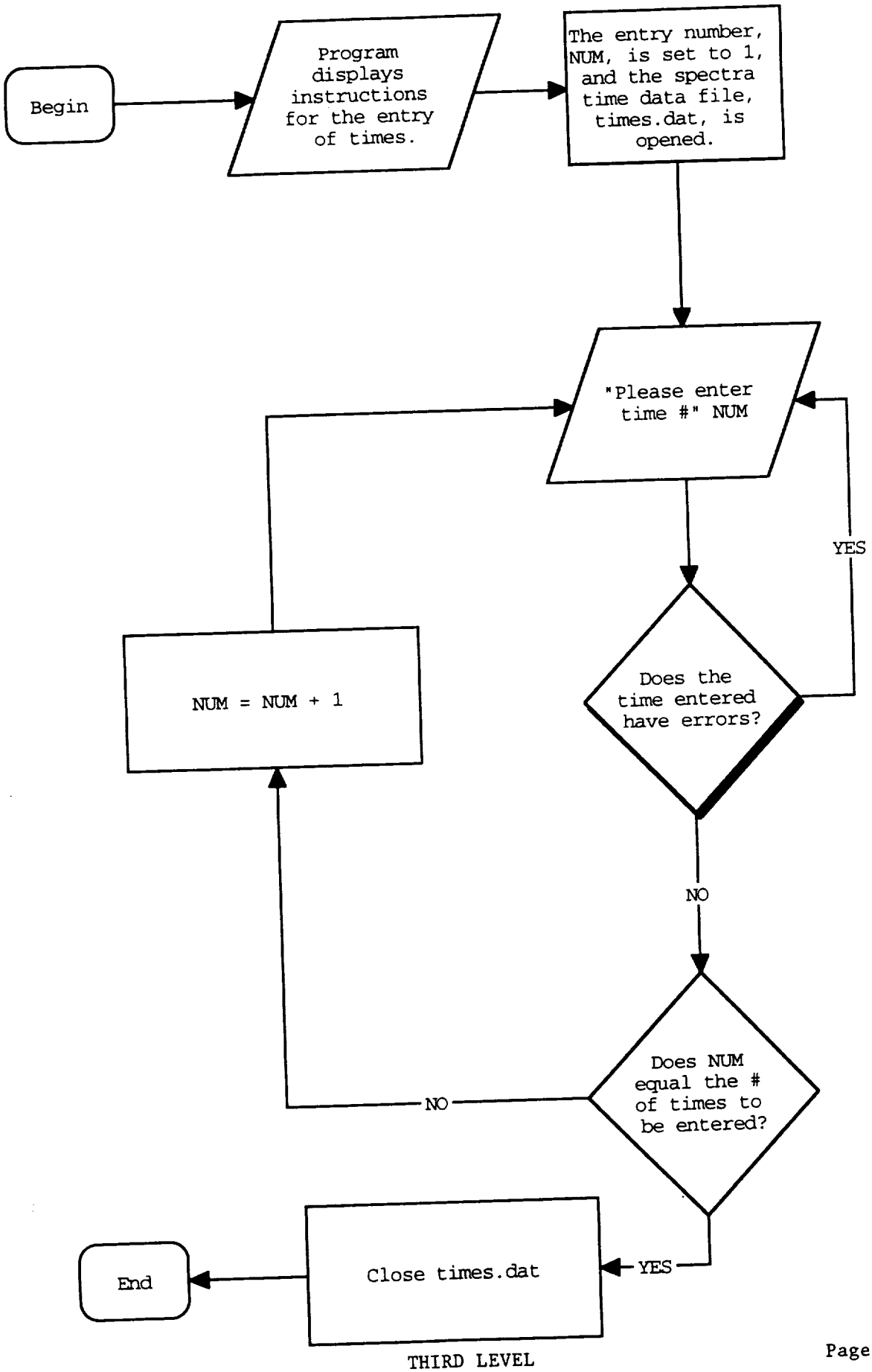

Begin
"FINISHED!!"
Program displays error.dat file.
Is the Ogilvie option enabled?
NO
YES
Has the option for data-transfer to remote sites been activated?
YES
NO
Ogilvie.dat is copied to the remote site.
Program displays Ogilvie.dat file.
End

User inputs program and output directories, plus RWNAST path and name. **Thu, Jul 21, 1994**



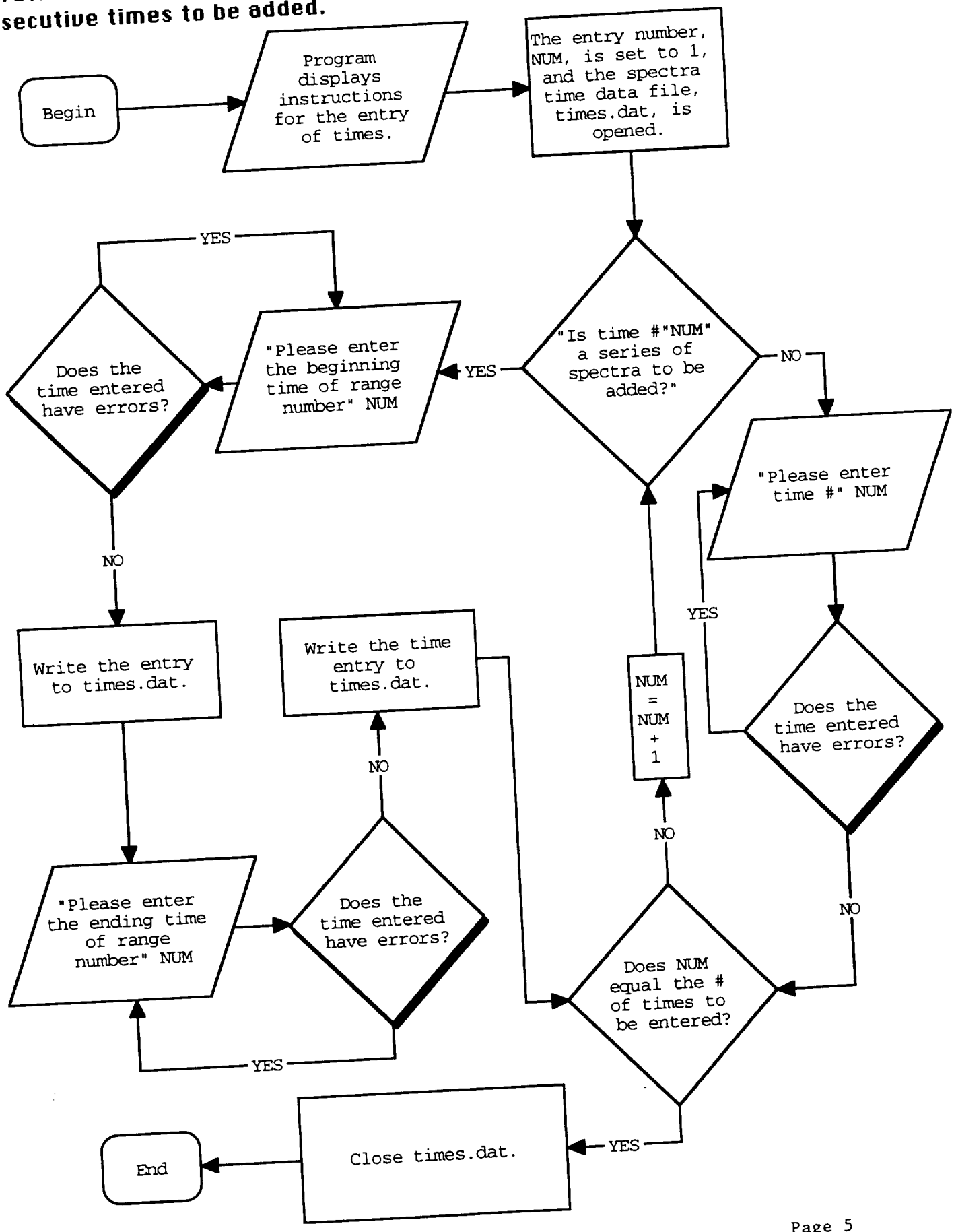

Begin
Operator is asked to enter the name of the remote site.
Operator is asked to enter the account name on the remote system.
Operator is asked to enter the account password (NO DEFAULT).
Operator is asked to enter the output directory on the remote system.
Were any entries left blank?
YES
"Blank entries are not allowed!"
NO
Current settings for site name, account, password, and remote directory are displayed.
"Are these settings correct?"
NO
YES
Program writes the definition of the remote site to the command file TEMP.COM
Program executes TEMP.COM (will now remember the definition for the rest of the session).
Program deletes TEMP.COM because the file contains the site's password, which is a security risk.
Program saves the new values for site name, account, and directory (NOT PASSWORD) to remote_site.dat
End

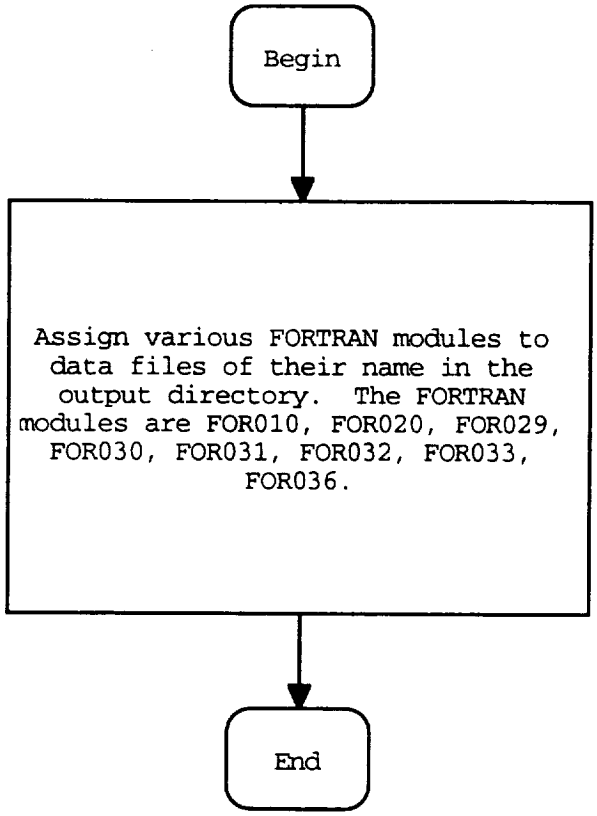

Begin
Read settings.dat file to determine the settings used the last time the program was run. These will be the default settings for ICWIND this time.
Program displays the current ICWIND settings.
"Are these settings OK?"
NO
Operator is requested to re-enter all ICWIND settings
YES
Current settings are saved in settings.dat file, to be used as defaults next time the program is run.
End


Begin
Program displays instructions for the entry of times.
The entry number, NUM, is set to 1, and the spectra time data file, times.dat, is opened.
"Please enter time #" NUM
Does the time entered have errors?
YES
NO
NUM = NUM + 1
Does NUM equal the # of times to be entered?
NO
YES
Close times.dat
End

THIRD LEVEL

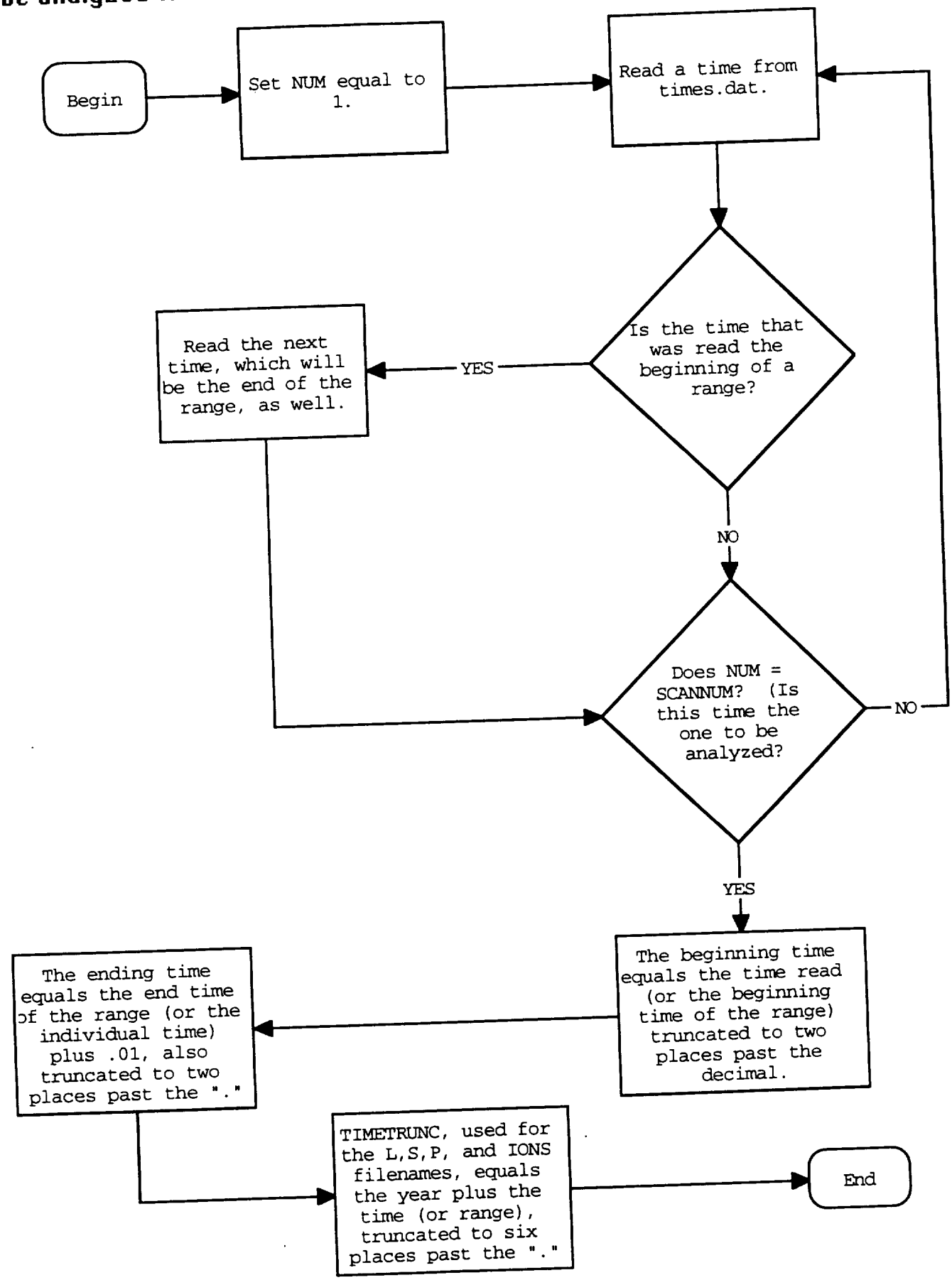
iperator enters individual times and
consecutive times to be added.




Begin
Assign various FORTRAN modules to data files of their name in the output directory. The FORTRAN modules are FOR010, FOR020, FOR029, FOR030, FOR031, FOR032, FOR033, FOR036.
End

Read the time corresponding to the spectra to be analyzed from times.dat.

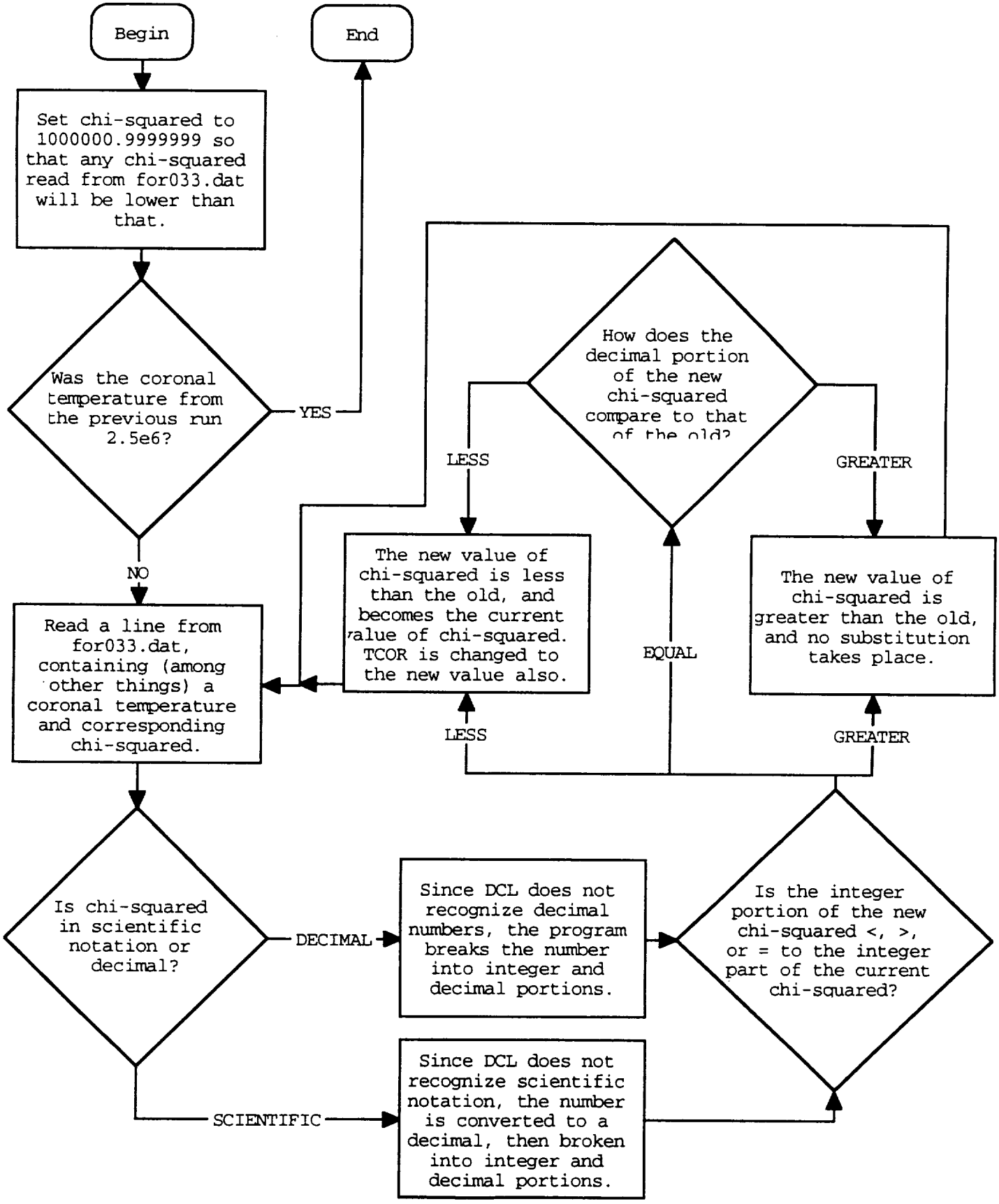
Thu, Jul 21, 1994



THIRD LEVEL

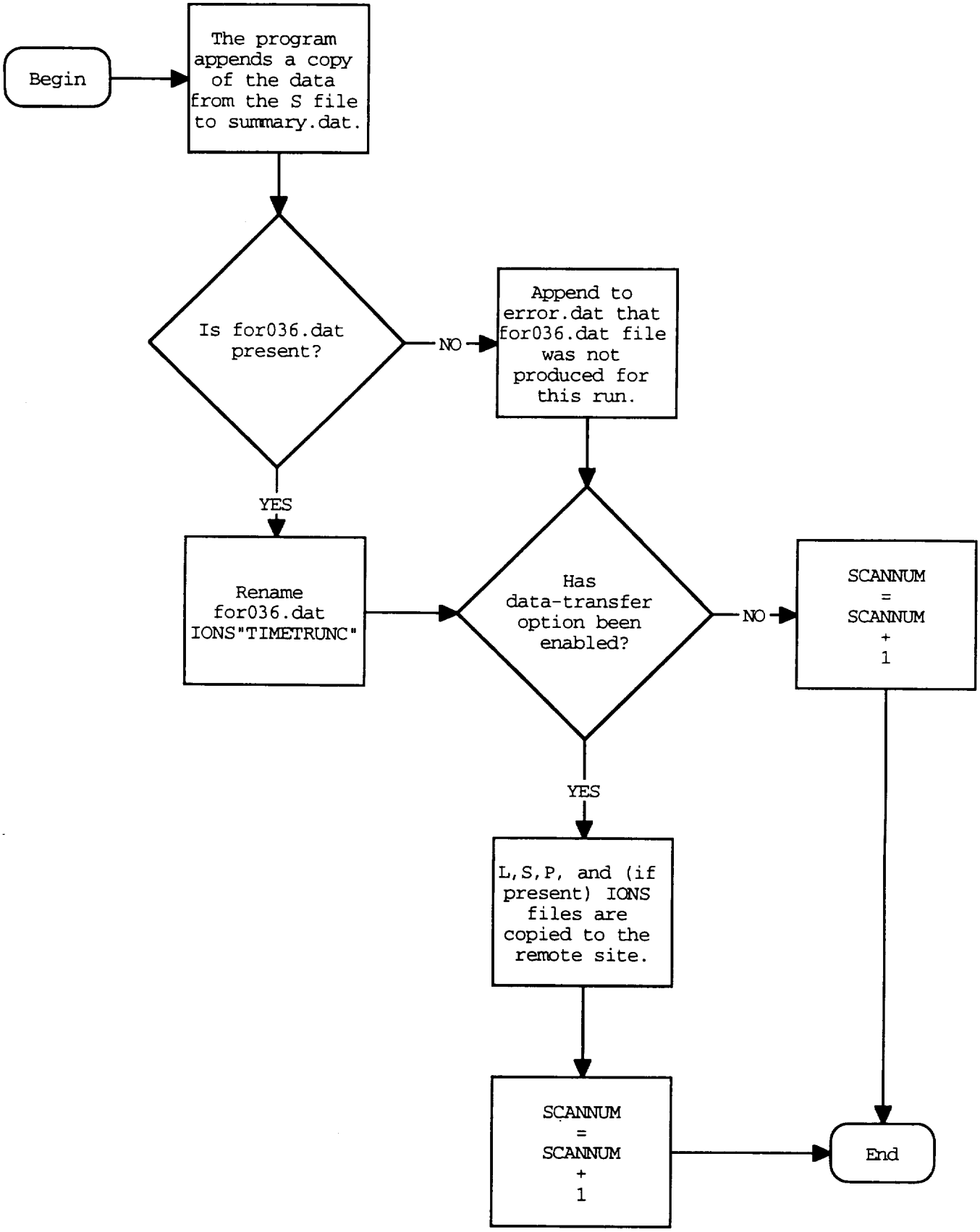
Check for033.dat file for coronal temperature corresponding to the lowest chi-squared.

Thu, Jul 21, 1994



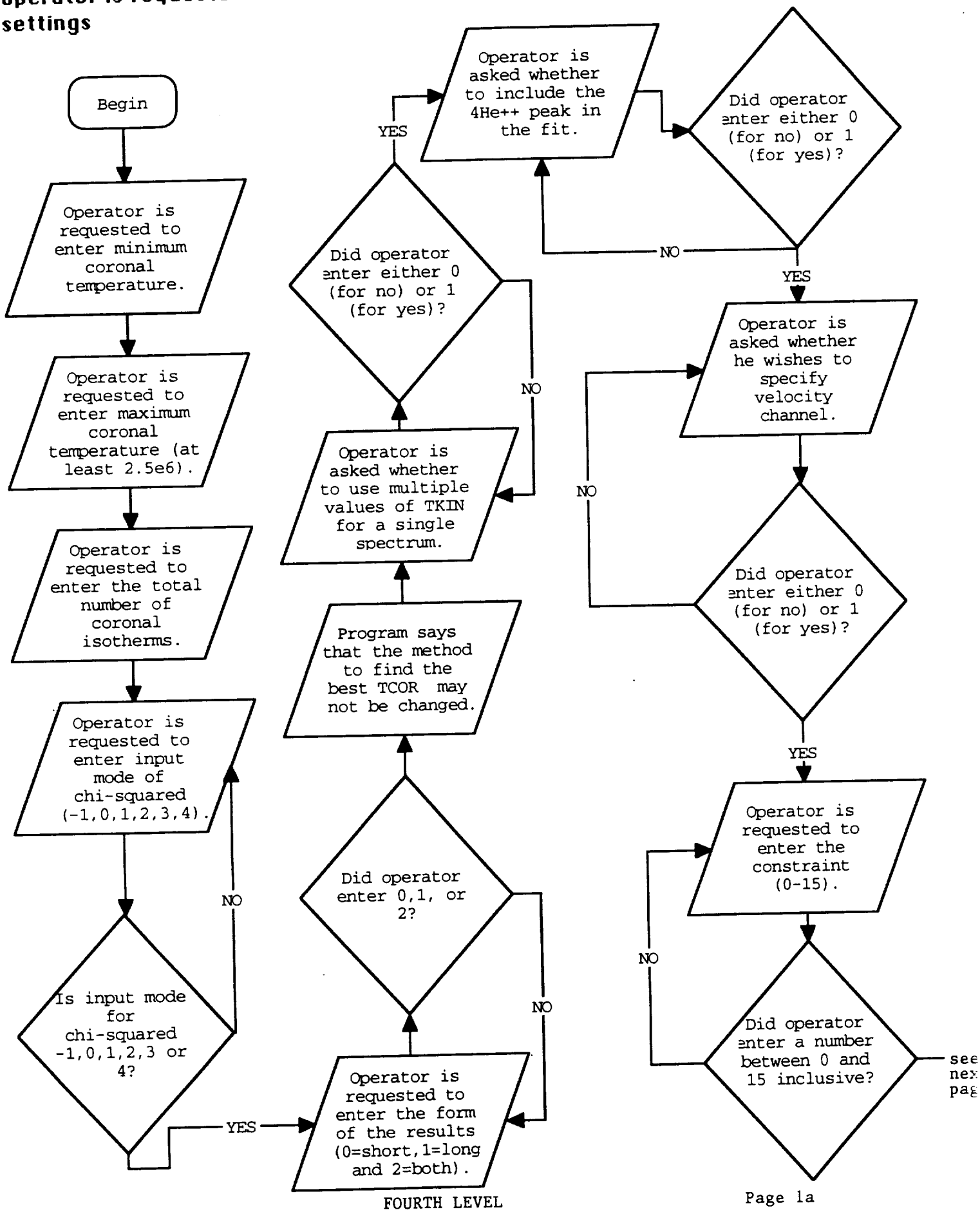
The program appends the data in the S file to summary.dat, renames for036.dat, and copies data files to remote site.

Thu, Jul 21, 1994

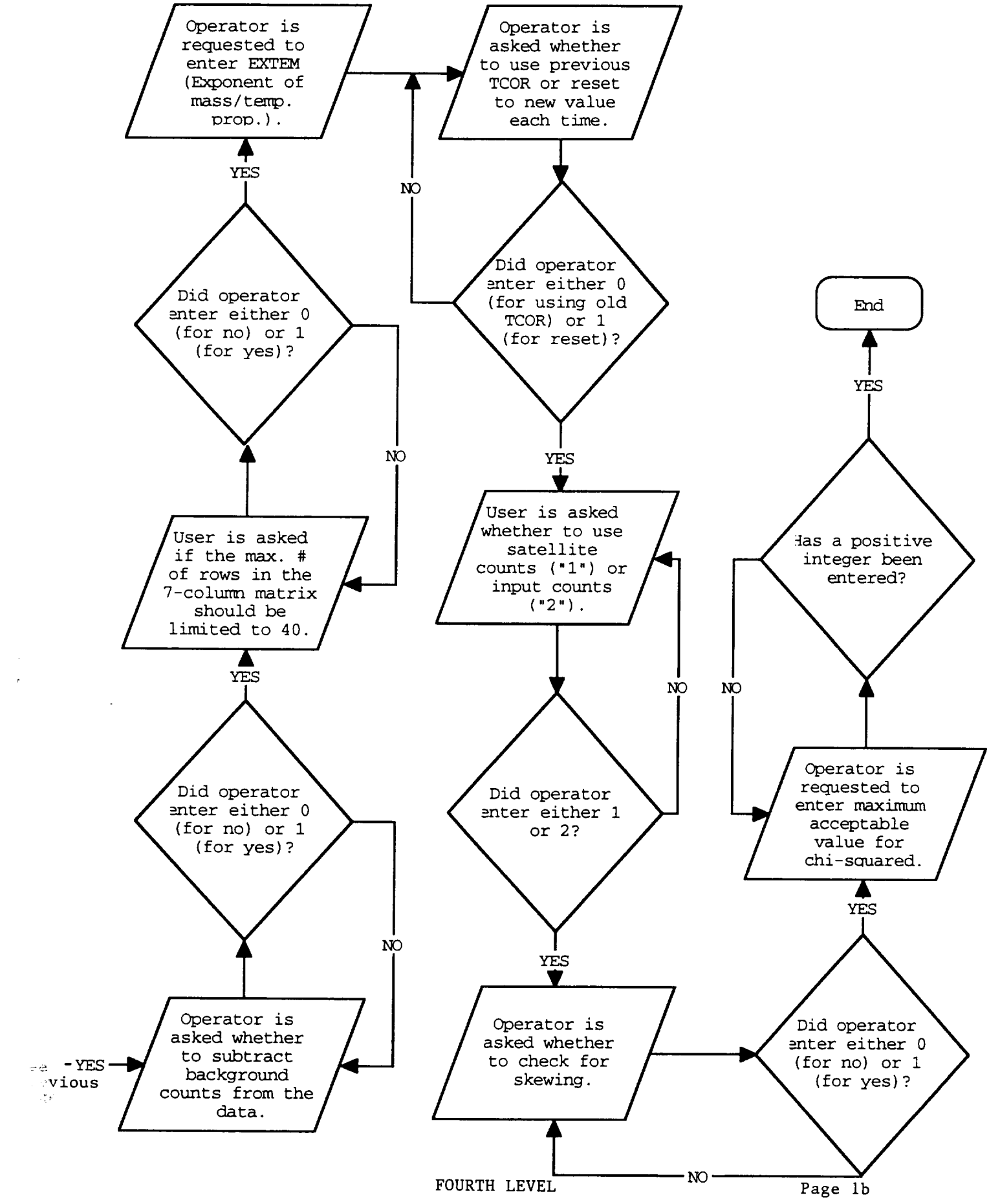


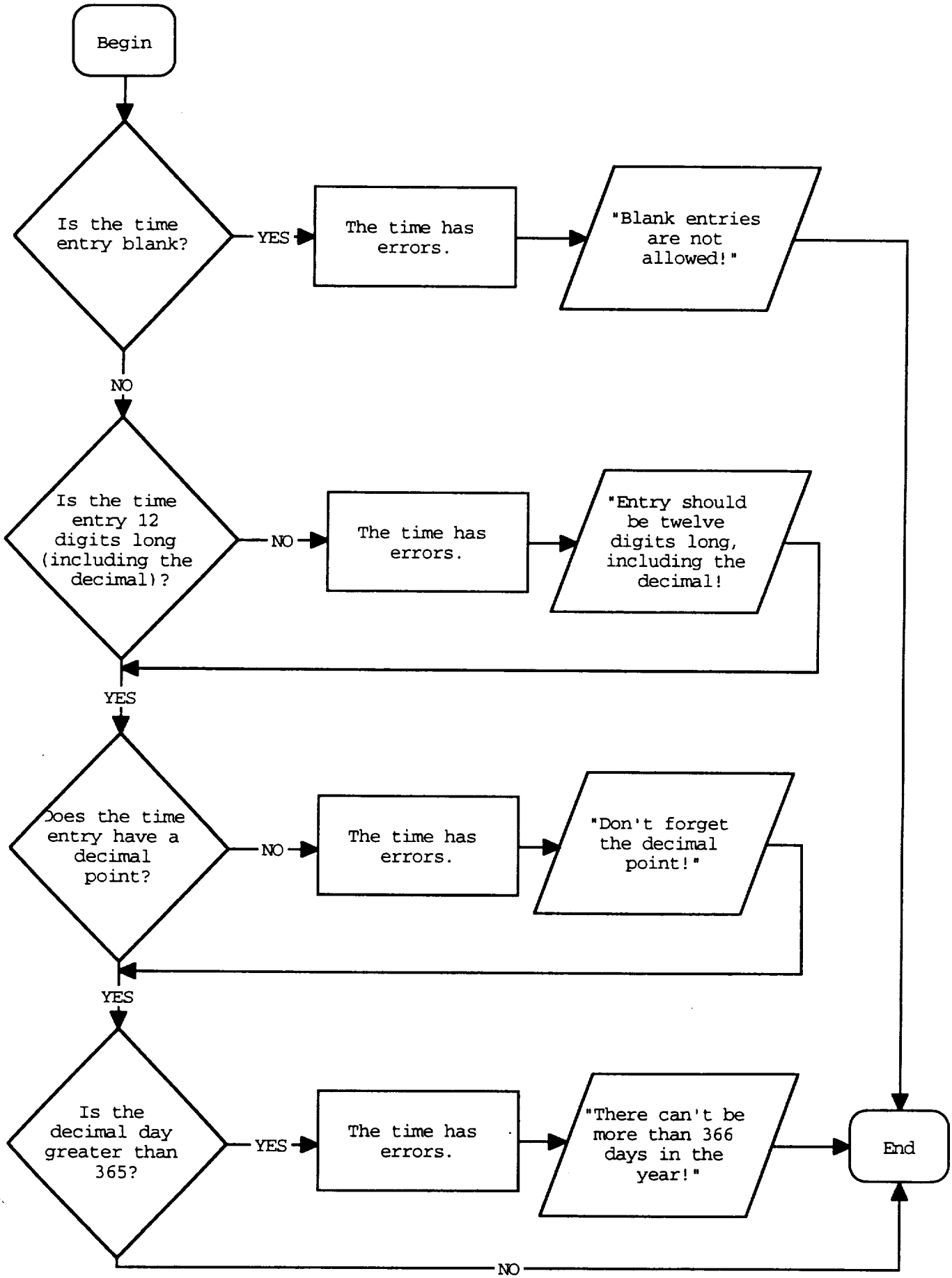
Operator is requested to re-enter all ICWIND settings

Thu, Jul 21, 1994



FOURTH LEVEL


Operator is requested to enter EXTEM (Exponent of mass/temp. prop.).
Operator is asked whether to use previous TCOR or reset to new value each time.
YES
NO
Did operator enter either 0 (for no) or 1 (for yes)?
Did operator enter either 0 (for using old TCOR) or 1 (for reset)?
NO
YES
User is asked if the max. # of rows in the 7-column matrix should be limited to 40.
User is asked whether to use satellite counts ("1") or input counts ("2").
End
YES
Has a positive integer been entered?
YES
NO
NO
Did operator enter either 0 (for no) or 1 (for yes)?
Did operator enter either 1 or 2?
Operator is requested to enter maximum acceptable value for chi-squared.
NO
YES
YES
Operator is asked whether to subtract background counts from the data.
Operator is asked whether to check for skewing.
Did operator enter either 0 (for no) or 1 (for yes)?
-YES
vious
NO
FOURTH LEVEL
Page 1b


Begin
Is the time entry blank?
YES
The time has errors.
"Blank entries are not allowed!"
NO
Is the time entry 12 digits long (including the decimal)?
NO
The time has errors.
"Entry should be twelve digits long, including the decimal!
YES
Does the time entry have a decimal point?
NO
The time has errors.
"Don't forget the decimal point!"
YES
Is the decimal day greater than 365?
YES
The time has errors.
"There can't be more than 366 days in the year!"
End
NO

```
$ !
$ ! Manager program for the ICWIND mass spectra analysis program
$ !     Batch programmer: Joseph Newhouse
$ !
$ !
$ ! Welcome prints a welcome message and some explanation of the default
$ ! answer function in the program.
$ !
$welcome:
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output "**********WELCOME TO ICWIND_KB_JDN!*************"
$ write sys$output ""
$ write sys$output ""
$ write sys$output ""
$ write sys$output "NOTE: Throughout this program, I will ask you questions that
 may look like this:"
$ write sys$output ""
$ write sys$output "What is your name [KEITH W. OGILVIE]?"
$ write sys$output ""
$ write sys$output "Whenever the question is followed by something enclosed in b
rackets, the item"
$ write sys$output "enclosed by the brackets is the DEFAULT ANSWER to the questi
on being asked."
$ write sys$output "If you only press ENTER (or RETURN) in response to the quest
ion, I will assume"
$ write sys$output "that the default answer is correct.  So if you pressed ENTER
 (or RETURN) in"
$ write sys$output "response to the question posed above, I would assume that yo
ur name is"
$ write sys$output "KEITH W. OGILVIE."
$ write sys$output ""
$ write sys$output "Now, on with the program!! Here is your first chance to use
the DEFAULT ANSWER"
$ write sys$output "feature of the program..."
$ write sys$output ""
$ !
$ ! Read_directories and set_directories find various directories to
$ ! be used in the running of ICWIND.  The previous setting is the default
```

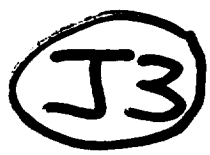
JOEY.COM

J1

```
$ ! setting, and the default setting is updated to match the new setting.
$ ! The data file used here is directories.dat.
$ !
$read_directories:
$ if F$SEARCH("directories.dat;*") .eqs. ""
$ then
$    oldprog_dir$ = ""
$    oldoutput_dir$ = ""
$    OLDRWNAST$ = ""
$    goto set_directories
$ endif
$ open/read DIR_READ directories.dat
$ read DIR_READ oldprog_dir$
$ read DIR_READ oldoutput_dir$
$ read DIR_READ OLDRWNAST$
$ close DIR_READ
$set_directories:
$ write sys$output "Please enter the directory where this file and the ICWIND pr
ogram files"
$ inquire/global prog_dir$ "are kept [''oldprog_dir$']"
$ if prog_dir$ .eqs. "" then prog_dir$ == oldprog_dir$
$ write sys$output "Please enter the output directory where I should generate th
e data"
$ inquire/global output_dir$ "files [''oldoutput_dir$']"
$ if output_dir$ .eqs. "" then output_dir$ == oldoutput_dir$
$ if output_dir$ .eqs. prog_dir$
$ then
$     write sys$output ""
$     write sys$output "Output directory cannot be the same as your program dire
ctory!"
$     write sys$output ""
$     goto set_directories
$ endif
$ inquire/global RWNAST$ "Please enter the name and path (directory) of the RWNA
ST data file to be used?  Default is ''OLDRWNAST$'"
$ if RWNAST$ .eqs. "" then RWNAST$ == OLDRWNAST$
$ if prog_dir$ .eqs. "" .or. output_dir$ .eqs. "" .or. RWNAST$ .eqs. ""
$ then
$    write sys$output ""
$    write sys$output "No blank entries are allowed!"
$    write sys$output ""
$ endif
$ write sys$output ""
$ write sys$output "The program directory is ''prog_dir$'"
$ write sys$output "The output directory for data files is ''output_dir$'"
$ write sys$output "The directory and name for the RWNAST data file is"
$ write sys$output "''RWNAST$'"
$ write sys$output ""
$ inquire ok "Are these settings correct [YES]?"
$ if ok .eqs. "" then ok = "yes"
$ if .not. ok then goto set_directories
$ open/write DIR_WRITE directories.dat
$ write DIR_WRITE prog_dir$
$ write DIR_WRITE output_dir$
$ write DIR_WRITE RWNAST$
$ close DIR_WRITE
$ !
$ ! Data_destination and destination will optionally set up automatic
$ ! transfers of the data files generated by the ICWIND program to a remote
$ ! site of the user's choice.  As usual, the default settings are the
$ ! settings of the program's previous use, and the defaults are then updated to
$ ! reflect the current settings, in the remote_site.dat file.
$ !
$data_destination:
$ write sys$output ""
$ inquire yes "Would you like to have the data transferred to a remote site [NO]
```

J2

```
"
$ if yes .eqs. "" then yes = "no"
$ if .not. yes
$ then
$    data_transfer=="no"
$    goto change_setting
$ endif
$ if F$SEARCH("remote_site.dat;*") .eqs. ""
$ then
$    OLDSITE$ = ""
$    OLDACCOUNT$ = ""
$    OLDDIRECTORY$ = ""
$    goto destination
$ endif
$ open/read SITE REMOTE_SITE.DAT
$ read SITE OLDSITE$
$ read SITE OLDACCOUNT$
$ read SITE OLDDIRECTORY$
$ close SITE
$destination:
$ inquire site$ "What is the name of the site [''OLDSITE$']?"
$ if site$ .eqs. "" then site$=OLDSITE$
$ inquire account$ "What is the name of the account to which the files should go
 [''OLDACCOUNT$']?"
$ if account$ .eqs. "" then account$ = OLDACCOUNT$
$ inquire password$ "What is the password of that account?"
$ write sys$output "To which directory, within that system, should the generated
 data files"
$ inquire directory$ "be sent [''OLDDIRECTORY$']?"
$ if directory$ .eqs. "" then directory$ = OLDDIRECTORY$
$ write sys$output ""
$ if site$ .eqs. "" .or. account$ .eqs. "" .or. password$ .eqs. "" .or. director
y$ .eqs. ""
$ then
$    write sys$output "No blank entries allowed!!"
$    write sys$output ""
$    goto destination
$ endif
$ write sys$output "So, the ICWIND data files should be sent to the account ''ac
count$' at"
$ write sys$output "the site ''site$' with the password ''password$', in the dir
ectory"
$ write sys$output "''directory$'"
$ write sys$output ""
$ inquire ok "Is this correct (y/n)?
$ if .not. ok then goto destination
$ open/write SITE2 REMOTE_SITE.DAT
$ write SITE2 site$
$ write SITE2 account$
$ write SITE2 directory$
$ close SITE2
$ open/write TEMP TEMP.COM
$ write TEMP "define UMD ''site$'""""""""'account$' ''password$'""""""::''directo
ry$'"
$ close TEMP
$ @TEMP
$ delete TEMP.COM;*
$ data_transfer == "yes"
$ !
$ ! Change_setting calls upon change_settings.com, a companion program to
$ ! joey.com, to change the settings used by the ICWIND program itself.  The
$ ! settings.dat file contains the old (default) settings.
$ !
$change_setting:
$ write sys$output ""
$ write sys$output "Let me show you the ICWIND program settings..."
```



```
$ @change_settings
$ write sys$output ""
$ !
$ ! Setup asks a couple of basic questions--how many times are to be analyzed
$ ! and from what year will they be.  Individual times and the addition of
$ ! consecutive times are counted the same way.  So, if you were to analyze
$ ! three individual times and two ranges of consecutive times (adding spectra),
$ ! you would enter the number 5 in answer to the first question.
$ !
$setup:
$ open/write SUMMARY 'output_dir$'summary.dat
$ write SUMMARY ""
$ close SUMMARY
$ write sys$output "Please note that this program counts individual times and ra
nges of times"
$ write sys$output "in the same way.  So, when you are asked the number of times
 to be analyzed,"
$ write sys$output "you should enter the SUM of the number of individual times a
nd the number of"
$ write sys$output "ranges of times."
$ write sys$output ""
$ write sys$output "How many different times (individual spectra) or ranges of c
onsecutive times"
$ inquire/global CHNUM "(where the corresponding spectra are added) would you li
ke to analyze?"
$check_chnum:
$ if CHNUM .lts. "0" .or. CHNUM .nes. F$STRING(F$INTEGER(CHNUM))
$ then
$    write sys$output ""
$    inquire CHNUM "You must enter a positive integer.  Please try again."
$    goto check_chnum
$ endif
$ inquire/global YR "What year are you studying (last two digits)?  19__"
$check_yr:
$ if YR .lts. "00" .or. F$LENGTH(YR) .ne. 2 .or. YR .nes. F$STRING(F$INTEGER(YR)
) .or. F$INTEGER(YR) .eq. 0
$ then
$    write sys$output ""
$    inquire YR "Please re-enter the last two digits of the year under considera
tion"
$    goto check_yr
$ endif
$ CHNUM==F$INTEGER(CHNUM) ! Converts CHNUM from string to integer
$ YR==F$INTEGER(YR) ! Converts YR to year
$ !
$ ! The following section is fairly self-explanatory.  Dr. Ogilvie suggested
$ ! I include a section of the program that would allow it to be run only from
$ ! input_scan.dat, as a way of getting a quick assessment of coronal
$ ! temperatures and chi-squared measurements.  As one who likes to give credit
$ ! where credit is due, I named the whole process the "Ogilvie option."
$ ! Besides, it has a certain ring to it...
$ !
$ write sys$output "So you will be studying ''CHNUM' times from 19''YR'."
$ inquire ok "Is this correct [YES]?"
$ if ok .eqs. "" then ok = "yes"
$ if .not. ok then goto setup
$Ogilvie:
$ write sys$output ""
$ inquire Ogilvie "Would you like to invoke the Ogilvie option [HELP]?"
$ if Ogilvie .eqs. "" then Ogilvie = "help"
$ if Ogilvie .eqs. "help"
$ then
$    write sys$output ""
$    write sys$output "The Ogilvie option allows you to run ICWIND from input_sc
an.dat alone.  The"
$    write sys$output "records the coronal temperature corresponding to the lowe
```

J4

```
st chi-squared,"
$    write sys$output "along with that value of chi-squared and the relevant dat
e, in the file"
$    write sys$output "OGILVIE.DAT.  This is an expeditious way to examine the d
ata to see if coronal"
$    write sys$output "temperatures indicate the presence of a coronal hole."
$    write sys$output ""
$    write sys$output "BEWARE!  Enabling the Ogilvie option (so named because of
 the man who suggested"
$    write sys$output "it) will cause the program NOT to produce any usable L, S
, P or ION files, since"
$    write sys$output "ICWIND is never run from input_file.dat."
$    write sys$output ""
$    inquire Ogilvie "Having read all that, would you like to invoke the Ogilvie
 option [NO]?"
$    if Ogilvie .eqs. "" then Ogilvie = "no"
$ endif
$ write sys$output ""
$ if Ogilvie
$ then
$    write sys$output "Ogilvie option on!"
$    open/write OGILVIE ogilvie.dat
$    write OGILVIE "Day/time            CHI-SQUARED   CORONAL TEMPERATURE"
$    write OGILVIE ""
$    close OGILVIE
$ endif
$ if .not. Ogilvie then write sys$output "Ogilvie option off!"
$ write sys$output ""
$ !
$ ! Analysis is a loop to determine whether all of the analyses will be
$ ! of single spectra or whether in some cases consecutive spectra will
$ ! be added.
$ !
analysis:
$ inquire/global ANALYSIS$ "Will you be adding consecutive mass spectra in any o
f your analyses (y/n)?"
$ if ANALYSIS$ .eqs. "Y" then goto data_entry2
$ if ANALYSIS$ .eqs. "N" then goto data_entry1
$ if ANALYSIS$ .eqs. "MFGOPLAK" then goto scan
$ write sys$output ""
$ write sys$output "You must enter 'y' or 'n'--try again."
$ goto analysis
$ !
$ ! Data_entry1 is the beginning of the process for entering times, where
$ ! only individual ion spectra will be analyzed.
$ !
$data_entry1:
$ write sys$output ""
$ NUM==1
$ write sys$output "You will be entering single times of mass spectra (e.g. 306.
39926481)."
$ write sys$output "If the day is less than 100, enter leading zeroes so there"
$ write sys$output "are still three digits before the decimal point."
$ write sys$output ""
$ write sys$output "NOTE: This program generates a summary file containing the d
ata from the"
$ write sys$output "S file for each run, for use in graphical analysis or spread
sheets.  If you wish"
$ write sys$output "to use this file, you MUST enter the times in CHRONOLOGICAL
order!!  For all"
$ write sys$output "other purposes, you may enter the times in any order you wis
h."
$ open/write TIME_FILE TIMES.DAT
$ !
$ ! The following is a loop for the user to enter the times to be analyzed.
$ ! The times are written to a file after they are checked for errors.
```

```
$ ! When the user has entered the number of times he specified at the
$ ! beginning, the program closes the file.
$ !
$save_times1:
$ write sys$output ""
$ inquire TIME$ "Please enter time #''NUM'"
$ gosub time_errors
$ if error .ge. 1 then goto save_times1
$ write TIME_FILE TIME$
$ if NUM .eq. CHNUM then goto close_times1
$ NUM==NUM+1
$ goto save_times1
$ !
$ ! The following is a subroutine designed to check that the times entered
$ ! have the correct number of digits with the decimal in the correct
$ ! place.  It also checks that the number of the day does not exceed 365,
$ ! since there cannot be more than 365 decimal days in a (leap) year.  It
$ ! also guards against accidental depression of the enter key.
$ ! Both data_entry1 and data_entry2 (as well as the time_range) make use of
$ ! this.
$ !
$time_errors:
$ error=0
$ if TIME$ .eqs. ""
$ then
$    write sys$output "Blank entries not allowed."
$    error=error+1
$    return
$ endif
$ if F$LENGTH(TIME$) .ne. 12
$ then
$    write sys$output "Entry should be 12 digits long, including the decimal poi
nt (e.g. 209.12435291)."
$    error=error+1
$ endif
$ if F$EXTRACT(3,1,TIME$) .nes. "."
$ then
$    write sys$output "Don't forget the decimal point!"
$    error=error+1
$ endif
$ if F$EXTRACT(0,3,TIME$) .gts. "365"
$ then
$    write sys$output "There can't be more than 366 days in the year!"
$    error=error+1
$ endif
$ if error .ge. 1 then write sys$output "Please re-enter number carefully."
$ return
$close_times1:
$ close TIME_FILE
$ goto time_check
$ !
$ ! Data_entry2 is the same as data_entry1 except it allows the user to
$ ! enter RANGES of times (i.e. beginning and ending times of a range) to
$ ! add consecutive mass spectra.
$ !
$data_entry2:
$ write sys$output ""
$ NUM==1
$ write sys$output "You will be entering single times or time ranges of mass spe
ctra"
$ write sys$output "(e.g. 306.39926481).  For ranges of times, ICWIND will add t
ogether all mass"
$ write sys$output "spectra recorded during that time range, increasing the accu
racy of the"
$ write sys$output "analysis.  If you choose to enter a range of times, you will
 be asked for the"
```

JG

```
$ write sys$output "beginning and ending times of the range of consecutive spect
ra to be added.
$ write sys$output ""
$ write sys$output "For all entries, if the day is less than 100, enter leading
zeroes so there"
$ write sys$output "are still three digits before the decimal point."
$ write sys$output ""
$ write sys$output "NOTE: This program generates a summary file containing the d
ata from the"
$ write sys$output "S file for each run, for use in graphical analysis or spread
sheets.  If you wish"
$ write sys$output "to use this file, you MUST enter the times in CHRONOLOGICAL
order!!  For all"
$ write sys$output "other purposes, you may enter the times in any order you wis
h."
$ open/write TIME_FILE2 TIMES.DAT
$ !
$ ! The following is a loop for the user to enter the times to be analyzed.
$ !
$ ! There is also a subroutine for entering ranges of times to be analyzed.
$ !
$save_times2a:
$ write sys$output ""
$ inquire RANGE$ "Is time #''NUM' a range of time to add consecutive spectra [NO
]?"
$ if RANGE$ .eqs. "" then RANGE$="no"
$ if RANGE$ then goto time_range_beginning
$save_times2b:
$ write sys$output ""
$ inquire TIME$ "Please enter time #''NUM'"
$ gosub time_errors
$ if error .ge. 1 then goto save_times2b
$ write TIME_FILE2 TIME$
$ if NUM .eq. CHNUM then goto close_times2
$ NUM==NUM+1
$ goto save_times2a
$ !
$ ! Time_range_beginning is a routine for the user to enter the starting time
$ ! of a range of times.
$ !
$time_range_beginning:
$ write sys$output ""
$ inquire TIME$ "Please enter the starting time for time range #''NUM'"
$ gosub time_errors
$ if error .ge. 1 then goto time_range_beginning
$ TIME$ = TIME$ + "BEG"
$ write TIME_FILE2 TIME$
$ !
$ ! Time_range_ending allows the user to enter the ending time of a range of
$ ! times corresponding to consecutive mass spectra to be analyzed.
$ !
$time_range_ending:
$ write sys$output ""
$ inquire TIME$ "Please enter the ending time for time range #''NUM'"
$ gosub time_errors
$ if error .ge. 1 then goto time_range_ending
$ TIME$ = TIME$ + "END"
$ write TIME_FILE2 TIME$
$ if NUM .eq. CHNUM then goto close_times2
$ NUM==NUM+1
$ goto save_times2a
$close_times2:
$ close TIME_FILE2
$ goto time_check
$ !
$ ! Time_check, read_times, and stop_check are provided as a means by which
```

J7

```
$ ! the user may verify that, in fact, he entered the times to be analyzed
$ ! correctly.  This is the last stop before the point of no return at which
$ ! the program takes over entirely and runs ICWIND on all the spectra.
$ ! The times are displayed and the user is asked to confirm them.
$ !
$time_check:
$ open/read CHECK TIMES.DAT
$ NUM==1
$ write sys$output ""
$read_times:
$ read CHECK CHECK$
$ if F$LENGTH(CHECK$) .gt. 12
$ then
$    if F$EXTRACT(12,3,CHECK$) .eqs. "BEG"
$    then
$       write sys$output "Starting time of range #''NUM' is ", F$EXTRACT(0,12,CH
ECK$)
$       goto read_times
$    endif
$    if F$EXTRACT(12,3,CHECK$) .eqs. "END"
$    then
$       write sys$output "Ending time of range #''NUM' is ", F$EXTRACT(0,12,CHEC
K$)
$    endif
$  else write sys$output "Individual time #''NUM' is ", CHECK$
$ endif
$ write sys$output ""
$ if NUM .eq. CHNUM then goto stop_check
$ NUM==NUM+1
$ goto read_times
$stop_check:
$ close CHECK
$ write sys$output ""
$ inquire ok "Are these times OK [YES]?"
$ if ok .eqs. "" then ok = "yes"
$ if .not. ok
$ then
$    write sys$output ""
$    write sys$output "In that case I will ask you to re-enter the times..."
$    write sys$output ""
$    goto analysis
$ endif
$ write sys$output "GREAT!  We're all set to run the program.  Just sit back and
 relax..."
$ write sys$output ""
$ goto scan
$ !
$ ! Scan runs icwind2.com, which is a modification of the original icwind.com,
$ ! to send the for___.dat files to the output directory of the user's choice
$ ! instead of always to weekly$:[weekly.ummtb].  An error file is also
$ ! established here so that future error messages may be appended to it.
$ ! I hope there won't be too many of those!  :)
$ !
$scan:
$ write sys$output "Setting up program to run..."
$ @icwind2
$ scannum == 1
$ open/write ERROR error.dat
$ write ERROR ""
$ close ERROR
$open_datafile:
 open/read SCANDATA times.dat
$ number = 1
$ !
$ ! Data_collection reads off the time of the spectra corresponding to the scan
$ ! number.  The data file is times.dat.  Various variables are set, such as
```

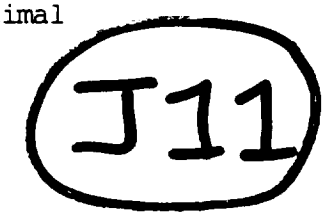
J8

```
$ ! TIME_BEG and TIME_END, which are the beginning and ending times entered in
$ ! the input_scan.dat file and the input_file.dat file.  TIMETRUNC is used
$ ! in the L, S, and P file names, and in the file name of the
$ ! ionsYR.DAY file (which is the FOR036.DAT file renamed--the FOR036.DAT file
$ ! is generated from the running of ICWIND_KB_JDN from the input_file.dat file.
$ !
$data_collection:
$ if scannum .gt. CHNUM then goto end_sequence
$ read SCANDATA data$
$ write sys$output data$
$ addzero="no"
$ if F$EXTRACT(12,3,data$) .eqs. "BEG"
$ then
$    read SCANDATA data2$
$    if number .eq. scannum
$    then
$       TIME_BEG == F$EXTRACT(0,6,data$)
$       day$ = F$EXTRACT(0,3,data2$)
$       decimal$ = F$EXTRACT(4,2,data2$)
$       if F$EXTRACT(4,1,data2$) .eqs. "0" then addzero="yes"
$       decimal = F$INTEGER(decimal$)
$       decimal = decimal + 1
$       decimal$ = F$STRING(decimal)
$       if decimal$ .eqs. "100"
$       then
$          decimal$ = "00"
$          day$ = F$STRING(F$INTEGER(day$) + 1)
$       endif
$       if addzero .and. decimal$ .nes. "10" then decimal$ = "0" + F$STRING(deci
mal)
$       TIME_END == day$ + "." + decimal$
$       if F$EXTRACT(2,1,data$) .eqs. F$EXTRACT(2,1,data2$) then TIMETRUNC == "'
'YR'." + (F$EXTRACT(0,7,data$) - ".") + "-" + F$EXTRACT(4,3,data2$)
$       if F$EXTRACT(2,1,data$) .nes. F$EXTRACT(2,1,data2$) then TIMETRUNC == "'
'YR'." + (F$EXTRACT(0,7,data$) - ".") + "-" + (F$EXTRACT(0,7,data2$) - ".")
$       write sys$output TIMETRUNC
$       close SCANDATA
$       goto write_input_scan
$    endif
$ endif
$ if number .eq. scannum
$ then
$    TIME_BEG == F$EXTRACT(0,6,data$)
$    day$ = F$EXTRACT(0,3,data$)
$    decimal$ = F$EXTRACT(4,2,data$)
$    if F$EXTRACT(4,1,data$) .eqs. "0" then addzero="yes"
$    decimal = F$INTEGER(decimal$)
$    decimal = decimal + 1
$    decimal$ = F$STRING(decimal)
$    if decimal$ .eqs. "100"
$    then
$       decimal$ = "00"
$       day$ = F$STRING(F$INTEGER(day$) + 1)
$    endif
$    if addzero .and. decimal$ .nes. "10" then decimal$ = "0" + F$STRING(decimal
)
$    TIME_END == day$ + "." + decimal$
$    TIMETRUNC == "'YR'." + (F$EXTRACT(0,7,data$) - ".")
$    write sys$output TIMETRUNC
$    close SCANDATA
$    goto write_input_scan
$ endif
$ number = number + 1
$ goto data_collection
$ !
$ ! Write_input_scan saves the relevant values (from change_settings.com) to
```

J9

```
$ ! input_scan.dat.  These values are parameters under which the ICWIND_KB_JDN
$ ! program might change.  First, though, the first and last times analyzed
$ ! are recorded for later use in the file name of the summary data file.
$ !
$write_input_scan:
$ if scannum .eq. 1 then TIME1==F$EXTRACT(0,7,data$) - "."
$ if scannum .eq. CHNUM then TIME2==F$EXTRACT(0,7,data$) - "."
$ open/write INPUTSCAN input_scan.dat
$ write INPUTSCAN isothermplotscan$
$ write INPUTSCAN mintemp$
$ write INPUTSCAN maxtemp$
$ write INPUTSCAN isothermnum$
$ write INPUTSCAN chisqr$
$ write INPUTSCAN resultform$
$ write INPUTSCAN tcor_method$
$ write INPUTSCAN mult_tkin$
$ write INPUTSCAN include_He$
$ write INPUTSCAN velocitychannel$
$ write INPUTSCAN constraint$
$ write INPUTSCAN background_counts$
$ write INPUTSCAN maxlimit$
$ write INPUTSCAN extem$
$ write INPUTSCAN satcounts$
$ write INPUTSCAN skewcheck$
$ write INPUTSCAN "''output_dir$'L''TIMETRUNC'"
$ write INPUTSCAN "''output_dir$'S''TIMETRUNC'"
$ write INPUTSCAN "''output_dir$'P''TIMETRUNC'"
$ write INPUTSCAN YR
$ write INPUTSCAN RWNAST$
$ write INPUTSCAN TIME_BEG
$ write INPUTSCAN TIME_END
$ write INPUTSCAN constraintweight$
$ close INPUTSCAN
$ !
$ ! Now, with icwind_inputscan, we actually run ICWIND_KB_JDN after assigning
$ ! the values in the input_scan.dat file to it.  Then the program changes to
$ ! the output directory.
$ !
$icwind_inputscan:
$ assign input_scan.dat for005
$ run icwind_kb_jdn
$ set default 'output_dir$'
$ !
$ ! Now the program must read the for033.dat file to determine which coronal
$ ! temp (TCOR) to use for the input_file.dat.  It looks for the TCOR
$ ! which is less than 2.5e6 K, and corresponds to the lowest value of
$ ! chi-squared.
$ !
$ ! Since the program always seeks lower values of chi-squared, I have set
$ ! the initial values higher than any self-respecting value of chi-squared
$ ! would condescend to be.
$ !
$open_for033:
$ open/read FOR for033.dat
$ integerchi = 1000000
$ decimalchi = 9999999
$ last = "no"
$ !
$ ! Now the program reads the file.  Two complicating factors: DCL (Digital
$ ! Command Language) refuses to acknowledge the existence of decimal numbers;
$ ! however, the ICWIND_KB_JDN program produces them in the for033.dat.  And,
$ ! when ICWIND_KB_JDN is not confounding DCL by producing decimal numbers, it
$ ! is flummoxing it by producing numbers in scientific notation.  (By the
$ ! way, DCL does not recognize scientific notation as a number, either.)
$ !
$ ! This means that, for the decimal numbers, I have to break each of them into
```

```
$ ! the number before the decimal and the number to the right of the decimal
$ ! if I want to compare them to other decimal numbers.  Scientific notation?
$ ! You don't want to know.  But if you do, head down to the scientific
$ ! notation section.
$ !
$read_for033:
$ if last then goto assign_tcor
$ read FOR input
$ newtcor$ = F$EXTRACT(3,3,input)
$ if newtcor$ .eqs. "2.5" then last = "yes"
$ newchi$ = F$EXTRACT(17,13,input)
$ if F$EXTRACT(26,1,input) .eqs. "E" then goto scientific_notation
$ decimal = F$LOCATE(".",newchi$)
$ newintegerchi$ = F$EXTRACT(0,decimal,newchi$)
$ newintegerchi = F$INTEGER(newintegerchi$)
$ newdecimalchi$ = F$EXTRACT((decimal + 1),(8 - decimal),newchi$)
$ newdecimalchi = F$INTEGER(newdecimalchi$)
$ !
$ ! Compare_chivalues is where the actual comparison of the chi-squared values
$ ! is performed.  Each value is compared to the current "low" value; if the
$ ! new value is lower than the current value, the new value becomes the
$ ! current value of chi-squared.
$ !
$ ! I use the following logic:  If the integer part of the new value of
$ ! chi-squared is HIGHER than the integer part of the current value of
$ ! chi-squared, then the new value of chi-squared has to be higher than the
$ ! current value of chi-squared, and no substitution takes place.  If the
$ ! integer part of the new value of chi-squared is LOWER than the integer
$ ! part of the current value of chi-squared, then the new value of chi-squared
$ ! has to be lower than the current value, so the new value of chi-squared
$ ! becomes the current value of chi-squared (is substituted in).  If the
$ ! integer parts of the new and current values of chi-squared are EQUAL, then
$ ! the decimal portions of the numbers must be compared, according to the
$ ! same logic.
$ !
$compare_chivalues:
$ if newintegerchi .gt. integerchi then goto read_for033
$ if newintegerchi .lt. integerchi
$ then
$    integerchi = newintegerchi
$    decimalchi = newdecimalchi
$    tcor$ == newtcor$
$    goto read_for033
$ endif
$ if newintegerchi .eq. integerchi
$ then
$    if newdecimalchi .ge. decimalchi then goto read_for033
$    if newdecimalchi .lt. decimalchi
$    then
$       decimalchi = newdecimalchi
$       tcor$ == newtcor$
$       goto read_for033
$    endif
$ endif
$ !
$ ! Scientific_notation accounts for the situation where ICWIND_KB_JDN decides
$ ! to be difficult and throws a chi-squared value in scientific notation into
$ ! the mix.
$ !
$ ! What is done here is to convert the number from scientific notation to
$ ! decimal notation, after which it can be broken into integer and decimal
$ ! portions, and compare_chivalues may be used.
$ !
$scientific_notation:
$ exponent$ = F$EXTRACT(10,3,newchi$)
$ exponent = F$INTEGER(exponent$)
```



```
$ if exponent .lt. 0
$ then
$    absexp = exponent*(-1)
$    rawdata$ = newchi$ - "."
$    newintegerchi = 0
$    if absexp .ge. 8
$    then
$       newdecimalchi = 0
$       goto compare_chivalues
$    endif
$    newdecimalchi$ = F$EXTRACT(0,(8-absexp),rawdata$)
$    newdecimalchi = F$INTEGER(newdecimalchi$)
$    goto compare_chivalues
$ endif
$ if exponent .gt. 0
$ then
$    rawdata$ = newchi$ - "."
$    if exponent .gt. 6 then goto read_for033
$    newintegerchi$ = F$EXTRACT(0,(exponent + 1),rawdata$)
$    newintegerchi = F$INTEGER(newintegerchi$)
$    newdecimalchi$ = F$EXTRACT((exponent + 1),(7 - exponent),rawdata$)
$    goto compare_chivalues
$ endif
$ if exponent .eq. 0
$ then
$    newintegerchi$ = F$EXTRACT(0,1,newchi$)
$    newintegerchi = F$INTEGER(newintegerchi$)
$    newdecimalchi$ = F$EXTRACT(2,7,newchi$)
$    newdecimalchi = F$INTEGER(newdecimalchi$)
$    goto compare_chivalues
$ endif
$ !
$ ! Assign_tcor checks to be sure that chi-squared is less than chisqr_max, the
$ ! maximum acceptable value of chisquared assigned in change_settings.com.  If
$ ! it is not, an error message is written to file and ionsYR.DAY is not
$ ! produced.
$ !
$ ! Also, "e6" is appended to the TCOR$ selected from for033.dat.
$ !
$assign_tcor:
$ close FOR
$ tcor$ == tcor$ + "e6"
$ if Ogilvie then goto write_Ogilvie
$ if integerchi .gt. chisqr_max
$ then
$    open/append ERROR 'prog_dir$'error.dat
$    write ERROR "For time ''data$', the minimum chi-squared value was ''integer
chi'."
$    write ERROR "Since chi-squared was greater than ''chisqr_max', ICWIND was n
ot run from input_file.dat."
$    write ERROR ""
$    close ERROR
$    scannum == scannum + 1
$    set default 'prog_dir$'
$    goto open_datafile
$ endif
$ write sys$output "Coronal temperature from FOR033.DAT file with lowest chi-squ
ared is ''tcor$'"
$ goto write_input_file
$write_Ogilvie:
$ set default 'prog_dir$'
  chi$ = F$STRING(integerchi) + "." + F$STRING(decimalchi)
$timetrunc_add:
$ if F$LENGTH(TIMETRUNC) .lt. 16
$ then
$    TIMETRUNC == TIMETRUNC + " "
```

J12

```
$    goto timetrunc_add
$ endif
$chi_add:
$ if F$LENGTH(chi$) .lt. 13
$ then
$    chi$ = chi$ + " "
     goto chi_add
$ endif
$ open/append OGILVIE ogilvie.dat
$ write OGILVIE "''TIMETRUNC'     ''chi$' ''tcor$'"
$ close OGILVIE
$ scannum == scannum + 1
$ goto open_datafile
$ !
$ ! Write_input_file saves relevant values from change_settings.com to the file
$ ! input_file.dat, used in the second running of ICWIND_KB_JDN in each cycle.
$ !
$write_input_file:
$ set default 'prog_dir$'
$ open/write INPUT_FILE input_file.dat
$ write INPUT_FILE isothermplotfile$
$ write INPUT_FILE chisqr$
$ write INPUT_FILE resultform$
$ write INPUT_FILE tcor_method$
$ write INPUT_FILE mult_tkin$
$ write INPUT_FILE include_He$
$ write INPUT_FILE velocitychannel$
$ write INPUT_FILE constraint$
$ write INPUT_FILE background_counts$
$ write INPUT_FILE maxlimit$
$ write INPUT_FILE extem$
$ write INPUT_FILE tcor$
$ write INPUT_FILE reset_tcor$
  write INPUT_FILE delta_tcor$
  write INPUT_FILE satcounts$
$ write INPUT_FILE skewcheck$
$ write INPUT_FILE "''output_dir$'L''TIMETRUNC'"
$ write INPUT_FILE "''output_dir$'S''TIMETRUNC'"
$ write INPUT_FILE "''output_dir$'P''TIMETRUNC'"
$ write INPUT_FILE YR
$ write INPUT_FILE RWNAST$
$ write INPUT_FILE TIME_BEG
$ write INPUT_FILE TIME_END
$ write INPUT_FILE constraintweight$
$ close INPUT_FILE
$ !
$ ! Icwind_inputfile runs ICWIND_KB_JDN under the parameters saved in the
$ ! input_file.dat file.  Then, it shifts to the output directory, where it
$ ! it renames the for036.dat file if it is present.  If it is not present,
$ ! an error message is saved to errors.dat.  If a remote site has been
$ ! selected to which to transfer the data, that is done now, too.
$ !
$icwind_inputfile:
$ assign input_file.dat for005
$ run icwind_kb_jdn
$ set def 'output_dir$'
$ purge
$ !
$ ! Summary, read_summaryfile, and close_summaryfile are designed to combine
$ ! the data contained within the individual S files into a single file which
$ ! then can be analyzed easily with spreadsheets or graphical software.
  ! Read_summaryfile searches for the line in each S file that begins with the
$ ! date, and appends it to a file which is eventually named
$ ! summary'YR'.'TIME1'-'TIME2', where TIME1 is the first time analyzed in the
$ ! series and TIME2 is the last time analyzed.
$ !
```

J13

```
$summary:
$ open/read SFILE s'TIMETRUNC'
$read_summaryfile:
$ read SFILE junk$
$ if F$EXTRACT(5,4,junk$) .eqs. F$STRING(F$INTEGER(F$EXTRACT(5,4,junk$))) .or. F
`EXTRACT(4,1,junk$) .eqs. "." then goto close_summaryfile
$ goto read_summaryfile
$close_summaryfile:
$ write sys$output junk$
$ close SFILE
$ open/append SUMMARY summary.dat
$ write SUMMARY junk$
$ close SUMMARY
$ !
$ ! Icwind_cleanup does many of the final book-keeping duties for each run,
$ ! such as finding and renaming the FOR036.dat file, and transferring data
$ ! to a remote site.
$ !
$icwind_cleanup:
$ if F$SEARCH("FOR036.DAT;*") .eqs. ""
$  then
$    open/append ERROR 'prog_dir$'error.dat
$    write ERROR "FOR036.DAT file not created for ''TIMETRUNC'!"
$    close ERROR
$    if data_transfer
$    then
$       copy L'TIMETRUNC' UMD
$       copy S'TIMETRUNC' UMD
$       copy P'TIMETRUNC' UMD
$    endif
$    scannum==scannum + 1
$    set default 'prog_dir$'
$    goto open_datafile
$ endif
$ rename for036.dat ions'TIMETRUNC'
$ if data_transfer
$ then
$    copy L'TIMETRUNC' UMD
$    copy S'TIMETRUNC' UMD
$    copy P'TIMETRUNC' UMD
$    copy ions'TIMETRUNC' UMD
$ endif
$ scannum==scannum + 1
$ set default 'prog_dir$'
$ goto open_datafile
$end_sequence:
$ rename 'output_dir$'summary.dat 'output_dir$'summary'YR'.'TIME1'-'TIME2'
$ if data_transfer then copy 'output_dir$'summary'YR'.'TIME1'-'TIME2' UMD
$ write sys$output ""
$ write sys$output "FFFFFFF  III   NNN    N   III    SSSSSS   H     H   EEEEEEE
  DDDDDD    !!"
$ write sys$output "F         I    N NN   N    I    S         H     H   E
  D     D   !!"
$ write sys$output "FFFF      I    N  NN  N    I     SSSSS    HHHHHHH   EEEE
  D     D   !!"
$ write sys$output "F         I    N   NN N    I          S   H     H   E
  D     D"
$ write sys$output "F        III   N    NNN   III   SSSSSS    H     H   EEEEEEE
  DDDDDD    !!
$ write sys$output ""
$ if Ogilvie then goto end_Ogilvie
  write sys$output "Errors encountered while running ICWIND were:"
$ type error.dat
$ write sys$output ""
$ goto More
$ !
```

J14

```
$ !
$ ! End_Ogilvie is to display the data in
$ ! the Ogilvie.dat file, created through use of the Ogilvie option to run
$ ! ICWIND only from the input_scan.dat file, and record the time, chi-squared,
$ ! and the coronal temperature.
  !
$end_Ogilvie:
$ if data_transfer then copy ogilvie.dat UMD
$ write sys$output "The Ogilvie option data is as follows:"
$ write sys$output ""
$ type ogilvie.dat
$ write sys$output ""
$More:
$ inquire more "Would you like to analyze more spectra (y/n)?"
$ if more
$ then
$    inquire ok "Are the current settings OK (y/n)?"
$    if ok then goto setup
$    goto change_setting
$ endif
$ purge
$ write sys$output ""
$ write sys$output "Thank you for using ICWIND, and have an ABSOLUTELY TERRIFIC
day!"
```

J15

```
$ !
$ ! This is a utility to change the parameters of the ICWIND program,
$ ! and is a companion program to JOEY.COM.  The two should not be
$ ! separated.
$ !
$ ! The program begins by reading the settings used the last time the program
$ ! was run.  These will be the default settings for the current run.
$ !
$ write sys$output ""
$ write sys$output ""
$ open/read CHANGE settings.dat
$ read CHANGE isothermplotscan$
$ read CHANGE isothermplotfile$
$ read CHANGE mintemp$
$ read CHANGE maxtemp$
$ read CHANGE isothermnum$
$ read CHANGE chisqr$
$ read CHANGE resultform$
$ read CHANGE tcor_method$
$ read CHANGE mult_tkin$
$ read CHANGE include_He$
$ read CHANGE velocitychannel$
$ read CHANGE constraint$
$ read CHANGE background_counts$
$ read CHANGE maxlimit$
$ read CHANGE extem$
$ read CHANGE reset_tcor$
$ read CHANGE delta_tcor$
$ read CHANGE satcounts$
$ read CHANGE skewcheck$
$ read CHANGE constraintweight$
$ read CHANGE chisqr_max$
$ close CHANGE
$ !
$ ! Display_settings prints the current settings on the screen and asks for
$ ! the user's approval.  If the user disapproves, the computer allows him
$ ! to change most of the settings.  We aim to please around here!
$ !
$display_settings:
$ write sys$output "Program plots isotherms for a single period when running fro
m input_scan.dat;"
$ write sys$output "It does not when running from input_file.dat.  DO NOT CHANGE
 THIS!!"
$ write sys$output "Minimum coronal temperature plotted: ''mintemp$'"
$ write sys$output "Maximum coronal temperature plotted: ''maxtemp$'"
$ write sys$output "Total number of isotherms: ''isothermnum$'"
$ write sys$output "Input mode for chi-squared: ''chisqr$'"
$ if chisqr$ .eqs. "-1" then write sys$output "           sum ((si-ci)**2/si)"
$ if chisqr$ .eqs. "0" then write sys$output "          sum ((si-ci)**2)"
$ if chisqr$ .eqs. "1" then write sys$output "          sum (((si-ci)/si)**2)"
$ if chisqr$ .eqs. "2" then write sys$output "          sum (abs (si-ci)/si)"
$ if chisqr$ .eqs. "3" then write sys$output "          sum (abs (si-ci))"
$ if chisqr$ .eqs. "4" then write sys$output "          (1/(NPTS-1))(sum ((si-ci
)**2))"
$ write sys$output "Form of results, where 0=short form, 1=long form, and 2=both
 forms: ''resultform$'"
$ write sys$output "Method of finding best coronal temperature, where 0=linear,
1 non-linear: ''tcor_method$'"
$ write sys$output "Use multiple values of TKIN for a single spectrum (0=no, 1=y
es): ''mult_tkin$'"
$ write sys$output "Include 4He++ peak in the fit (0=no, 1=yes): ''include_He$'"
$ write sys$output "Specify velocity channel used (0=no, 1=yes): ''velocitychann
el$'"
$ write sys$output "Constraint used: ''constraint$'"
```

CHANGE_SETTINGS.COM

C1

```
$ if constraint$ .eqs. "0" then write sys$output "      NO CONSTRAINTS"
$ if constraint$ .eqs. "1" then write sys$output "      Kunz: N/O=.125  Mg/C=.096
  Si/C=.09"
$ if constraint$ .eqs. "2" then write sys$output "      C/O=.6  Mg/C=.096  Si/C=.
09"
$ if constraint$ .eqs. "3" then write sys$output "      C+5/O=.13  Mg/C=.096  Si/
C  09"
$ if constraint$ .eqs. "4" then write sys$output "      C+5/O=.13  Si/C=.09"
$ if constraint$ .eqs. "5" then write sys$output "      N/O=.125  Mg/C=.096"
$ if constraint$ .eqs. "6" then write sys$output "      N/O=.125  Mg/C=.096  Si/N
=.7"
$ if constraint$ .eqs. "7" then write sys$output "      C/O=.45  N/O=.125  S/O=.0
5"
$ if constraint$ .eqs. "8" then write sys$output "      C/O=.45  N/O=.125  Mg/O=.
175"
$ if constraint$ .eqs. "9" then write sys$output "      C/O=.45  N/O=.125  Mg/Si=
1.0"
$ if constraint$ .eqs. "10" then write sys$output "      C/O=.45  N/O=.125  Mg/C=
.096  Si/C=.09"
$ if constraint$ .eqs. "11" then write sys$output "      C/O=1.5  N/O=.125  Mg/Si
=1.0"
$ if constraint$ .eqs. "12" then write sys$output "      C/O=1.0  N/O=.125  Mg/Si
=1.1"
$ if constraint$ .eqs. "13" then write sys$output "      C/O=.45  N/O=.125"
$ if constraint$ .eqs. "14" then write sys$output "      C/O=.45  N/O=.125  Ne/O=
.15"
$ if constraint$ .eqs. "15" then write sys$output "      C/O=.45  N/O=.125  Si/O=
.1"
$ write sys$output "Subtract background counts from the data (0=no, 1=yes): ''ba
ckground_counts$'"
$ write sys$output "Limit max. # of rows in 7-column data matrix to 40 (0=no, 1=
yes): ''maxlimit$'"
$  rite sys$output "EXTEM (Exponent of mass/temp. prop.): ''extem$'"
$ write sys$output "Use previous TCOR or reset to new value each time (0=use old
 TCOR, 1=reset): ''reset_tcor$'"
$ write sys$output "Delta TCOR: ''delta_tcor$'"
$ write sys$output "Use satellite counts (1) or input counts (2): ''satcounts$'"
$ write sys$output "Check for skewing (0=no, 1=yes): ''skewcheck$'"
$ write sys$output "Constraint weight: ''constraintweight$'"
$ write sys$output "Maximum acceptable value of chi-squared: ''chisqr_max$'"
$ write sys$output ""
$ inquire ok "Are these settings OK (y/n)?"
$ if ok then goto write_settings
$ !
$ ! Change_settings allows the user to change most of the settings.
$ !
$change_settings$:
$ inquire newmintemp$ "Minimum coronal temp [''mintemp$']"
$ if newmintemp$ .nes. "" then mintemp$=newmintemp$
$ inquire newmaxtemp$ "Maximum coronal temp (MUST BE AT LEAST 2.5e6!!) [''maxtem
p$']"
$ if newmaxtemp$ .nes. "" then maxtemp$=newmaxtemp$
$ inquire newisothermnum$ "Total number of coronal isotherms [''isothermnum$']"
$ if newisothermnum$ .nes. "" then isothermnum$=newisothermnum$
$ !
$ ! Chisqr makes sure that the user has entered -1, 0, 1, 2, 3, or 4 for
$ ! the input mode of chi-squared.
$ !
$ chisqr:
$ write sys$output "Possible modes for chi-squared:"
$ write sys$output "     -1    =          sum ((si-ci)**2/si)"
$ write sys$output "      0    =          sum ((si-ci)**2)"
$ write sys$output "      1    =          sum (((si-ci)/si)**2)"
$ write sys$output "      2    =          sum (abs (si-ci)/si)"
```

```
$ write sys$output "      3    =           sum (abs (si-ci))"
$ write sys$output "      4    =           (1/(NPTS-1))(sum ((si-ci)**2))"
$ write sys$output ""
$ inquire newchisqr$ "Input mode for chi-squared [''chisqr$']"
$ if newchisqr$ .nes. "" .and. newchisqr$ .nes. "-1" .and. newchisqr$ .nes. "0"
.and. newchisqr$ .nes. "1" .and. newchisqr$ .nes. "2" .and. newchisqr$ .nes. "3"
   .nd. newchisqr$ .nes. "4"
$ then
$     write sys$output ""
$     write sys$output "Input mode must be -1, 0, 1, 2, 3, or 4!  Please try agai
n."
$     write sys$output ""
$     goto chisqr
$ endif
$ if newchisqr$ .nes. "" then chisqr$=newchisqr$
$ !
$ ! Resultform confirms that the user enters 0, 1, or 2 for the form of the
$ ! results.
$ !
$resultform:
$ inquire newresultform$ "Form of results, where 0=short form, 1=long form, and
2=both forms [''resultform$']"
$ if newresultform$ .nes. "" .and. newresultform$ .nes. "0" .and. newresultform$
  .nes. "1" .and. newresultform$ .nes. "2"
$ then
$     write sys$output ""
$     write sys$output "You must enter 0, 1, or 2!  Please try again."
$     write sys$output ""
$     goto resultform
$ endif
$ if newresultform$ .nes. "" then resultform$=newresultform$
$ !
$   The method of finding the best coronal temperature is not allowed to be
$ ! changed because it does not need to be and it changes the ICWIND prompts.
$ !
$ write sys$output "Method of finding best coronal temperature, where 0=linear,
1=non-linear [''tcor_method$']"
$ write sys$output "This cannot be changed!"
$ write sys$output ""
$ !
$ ! Again we check to be sure that the user enters 0 or 1.
$ !
$mult_tkin:
$ inquire newmult_tkin$ "Use multiple values of TKIN for a single spectrum (0=no
, 1=yes) [''mult_tkin$']"
$ if newmult_tkin$ .nes. "" .and. newmult_tkin$ .nes. "0" .and. newmult_tkin$ .n
es. "1"
$ then
$     write sys$output ""
$     write sys$output "You must enter either 0 or 1.  Please try again."
$     write sys$output ""
$     goto mult_tkin
$ endif
$ if newmult_tkin$ .nes. "" then mult_tkin$=newmult_tkin$
$include_He:
$ inquire newinclude_He$ "Include 4He++ peak in the fit (0=no, 1=yes) [''include
_He$']"
$ if newinclude_He$ .nes. "" .and. newinclude_He$ .nes. "0" .and. newinclude_He$
  .nes. "1"
$ then
$     write sys$output ""
$     write sys$output "You must enter either 0 or 1.  Please try again."
$     write sys$output ""
$     goto include_He
```

C3

```
$ endif
$ if newinclude_He$ .nes. "" then include_He$=newinclude_He$
$velocitychannel:
$ inquire newvelocitychannel$ "Do you wish to specify velocity channel used (0=n
o, 1=yes) [''velocitychannel$']"
$ if newvelocitychannel$ .nes. "" .and. newvelocitychannel$ .nes. "0" .and. newv
e'>citychannel$ .nes. "1"
$ then
$     write sys$output ""
$     write sys$output "You must enter either 0 or 1.  Please try again."
$     write sys$output ""
$     goto velocitychannel
$ endif
$ if newvelocitychannel$ .nes. "" then velocitychannel$=newvelocitychannel$
$ write sys$output "Present constraint used: ''constraint$'.  Possible constrain
ts:"
$ write sys$output "      0:   NO CONSTRAINTS"
$ write sys$output "      1:   Kunz constraints: N/O=.125  Mg/C=.096  Si/C=.09"
$ write sys$output "      2:   C/O=.6  Mg/C=.096  Si/C=.09"
$ write sys$output "      3:   C+5/O=.13  Mg/C=.096  Si/C=.09"
$ write sys$output "      4:   C+5/O=.13  Si/C=.09"
$ write sys$output "      5:   N/O=.125  Mg/C=.096"
$ write sys$output "      6:   N/O=.125  Mg/C=.096  Si/N=.7"
$ write sys$output "      7:   C/O=.45  N/O=.125  S/O=.05"
$ write sys$output "      8:   C/O=.45  N/O=.125  Mg/O=.175"
$ write sys$output "      9:   C/O=.45  N/O=.125  Mg/Si=1.0"
$ write sys$output "      10:  C/O=.45  N/O=.125  Mg/C=.096  Si/C=.09"
$ write sys$output "      11:  C/O=1.5  N/O=.125  Mg/Si=1.0"
$ write sys$output "      12:  C/O=1.0  N/O=.125  Mg/Si=1.1"
$ write sys$output "      13:  C/O=.45  N/O=.125"
$ write sys$output "      14:  C/O=.45  N/O=.125  Ne/O=.15"
$ write sys$output "      15:  C/O=.45  N/O=.125  Si/O=.1"
$
$ ! Constraint constrains the user to enter an integer between 0 and 15 for
$ ! the constraint.
$ !
$constraint:
$ write sys$output ""
$ inquire newconstraint$ "Which constraint would you like to use (0-15) [''const
raint$']"
$ if newconstraint$ .gts. "15" .or. (newconstraint$ .nes. F$STRING(F$INTEGER(new
constraint$)) .and. newconstraint$ .nes. "") .or. (newconstraint$ .lts. "0" .and
. newconstraint$ .nes. "")
$ then
$     write sys$output "You must enter an integer between 0 and 15.  Please try a
gain."
$     goto constraint
$ endif
$ if newconstraint$ .nes. "" then constraint$=newconstraint$
$ !
$ ! There are many more instances where the computer checks to be sure that
$ ! the user has entered a correct value.
$ !
$background_counts:
$ inquire newbackground_counts$ "Subtract background counts from the data (0=no,
 1=yes) [''background_counts$']"
$ if newbackground_counts$ .nes. "" .and. newbackground_counts$ .nes. "0" .and.
>wbackground_counts$ .nes. "1"
 hen
     write sys$output ""
     write sys$output "You must enter either 0 or 1.  Please try again."
     write sys$output ""
$     goto background_counts
$ endif
```

C4

```
$ if newbackground_counts$ .nes. "" then background_counts$=newbackground_counts
$
$maxlimit:
$ inquire newmaxlimit$ "Limit max. # of rows in 7-column data matrix to 40 (0=no
, 1=yes) [''maxlimit$']"
$ if newmaxlimit$ .nes. "" .and. newmaxlimit$ .nes. "0" .and. newmaxlimit$ .nes.
   "
$ then
$     write sys$output ""
$     write sys$output "You must enter either 0 or 1.  Please try again."
$     write sys$output ""
$     goto maxlimit
$ endif
$ if newmaxlimit$ .nes. "" then maxlimit$=newmaxlimit$
$ inquire newextem$ "EXTEM (Exponent of mass/temp. prop.) [''extem$']"
$ if newextem$ .nes. "" then extem$=newextem$
$reset_tcor:
$ inquire newreset_tcor$ "Use previous TCOR or reset to new value each time (0=u
se old TCOR, 1=reset) [''reset_tcor$']"
$ if newreset_tcor$ .nes. "" .and. newreset_tcor$ .nes. "0" .and. newreset_tcor$
 .nes. "1"
$ then
$     write sys$output ""
$     write sys$output "You must enter either 0 or 1.  Please try again."
$     write sys$output ""
$     goto reset_tcor:
$ endif
$ if newreset_tcor$ .nes. "" then reset_tcor$=newreset_tcor$
$ inquire newdelta_tcor$ "Delta TCOR [''delta_tcor$']"
$ if newdelta_tcor$ .nes. "" then delta_tcor$=newdelta_tcor$
$satcounts:
$ inquire newsatcounts$ "Use satellite counts (1) or input counts (2)? [''satcou
1  .$']"
$ if newsatcounts$ .nes. "" .and. newsatcounts$ .nes. "1" .and. newsatcounts$ .n
es. "2"
$ then
$     write sys$output ""
$     write sys$output "You must enter either 1 or 2.  Please try again."
$     write sys$output ""
$     goto satcounts
$ endif
$ if newsatcounts$ .nes. "" then satcounts$=newsatcounts$
$skewcheck:
$ inquire newskewcheck$ "Check for skewing (0=no, 1=yes) [''skewcheck$']"
$ if newskewcheck$ .nes. "" .and. newskewcheck$ .nes. "0" .and. newskewcheck$ .n
es. "1"
$ then
$     write sys$output ""
$     write sys$output "You must enter either 0 or 1.  Please try again."
$     write sys$output ""
$     goto skewcheck
$ endif
$ if newskewcheck$ .nes. "" then skewcheck$=newskewcheck$
$ inquire newconstraintweight$ "Constraint weight [''constraintweight$']?"
$ if newconstraintweight$ .nes. "" then constraintweight$ = newconstraintweight$
$chisqr_max:
$ inquire newchisqr_max$ "Maximum acceptable value of chi-squared (INTEGERS ONLY
'   [''chisqr_max$']?"
$ if newchisqr_max$ .nes. "" .and. (newchisqr_max$ .nes. F$STRING(F$INTEGER(newc
hisqr_max$)) .or. newchisqr_max$ .lts. "0")
$ then
$     write sys$output ""
$     write sys$output "You must enter a positive integer."
$     write sys$output ""
```

C5

```
$     goto chisqr_max
$ endif
$ if newchisqr_max$ .nes. "" then chisqr_max$=newchisqr_max$
$ inquire continue "Hit ENTER to continue."
$ write sys$output ""
$ goto display_settings
$ !
$ ! Once the user has confirmed the validity of the settings, they are written
$ ! to disk to be used as the defaults for next time.  They are also converted
$ ! into global variables so that the joey.com program may use them.
$ !
$write_settings:
$  isothermplotscan$==isothermplotscan$
$  isothermplotfile$==isothermplotfile$
$  mintemp$==mintemp$
$  maxtemp$==maxtemp$
$  isothermnum$==isothermnum$
$  chisqr$==chisqr$
$  resultform$==resultform$
$  tcor_method$==tcor_method$
$  mult_tkin$==mult_tkin$
$  include_He$==include_He$
$  velocitychannel$==velocitychannel$
$  constraint$==constraint$
$  background_counts$==background_counts$
$  maxlimit$==maxlimit$
$  extem$==extem$
$  reset_tcor$==reset_tcor$
$  delta_tcor$==delta_tcor$
$  satcounts$==satcounts$
$  skewcheck$==skewcheck$
$  constraintweight$==constraintweight$
$   hisqr_max==F$INTEGER(chisqr_max$)
$  open/write CHANGE settings.dat
$  write CHANGE isothermplotscan$
$  write CHANGE isothermplotfile$
$  write CHANGE mintemp$
$  write CHANGE maxtemp$
$  write CHANGE isothermnum$
$  write CHANGE chisqr$
$  write CHANGE resultform$
$  write CHANGE tcor_method$
$  write CHANGE mult_tkin$
$  write CHANGE include_He$
$  write CHANGE velocitychannel$
$  write CHANGE constraint$
$  write CHANGE background_counts$
$  write CHANGE maxlimit$
$  write CHANGE extem$
$  write CHANGE reset_tcor$
$  write CHANGE delta_tcor$
$  write CHANGE satcounts$
$  write CHANGE skewcheck$
$  write CHANGE constraintweight$
$  write CHANGE chisqr_max$
$  close CHANGE
$  write sys$output ""
```

C6